

TNO-rapport  
FEL-97-A253

## Software Specificaties BIAS

TNO Fysisch en Elektronisch  
Laboratorium

Oude Waalsdorperweg 63  
Postbus 96864  
2509 JG 's-Gravenhage

Telefoon 070 374 00 00  
Fax 070 328 09 61

Datum  
september 1997

Auteur(s)  
Ir. H.G.M. Bennis  
Ir. A.P.T.M. Onderwater

19980413 152

Rubricering  
Vastgesteld door : Ing. R.S.E. Huisman  
Vastgesteld d.d. : 17 september 1997

Titel : Ongerubriceerd  
Managementuittreksel : Ongerubriceerd  
Samenvatting : Ongerubriceerd  
Rapporttekst : Ongerubriceerd  
Bijlagen A - B : Ongerubriceerd

Alle rechten voorbehouden.  
Niets uit deze uitgave mag worden  
vermenigvuldigd en/of openbaar gemaakt  
door middel van druk, fotokopie, microfilm  
of op welke andere wijze dan ook, zonder  
voorafgaande toestemming van TNO.

Indien dit rapport in opdracht werd  
uitgebracht, wordt voor de rechten en  
verplichtingen van opdrachtgever en  
opdrachtnemer verwezen naar de  
Algemene Voorwaarden voor onderzoeks-  
opdrachten aan TNO, dan wel de  
betreffende terzake tussen partijen  
gesloten overeenkomst.  
Het ter inzage geven van het TNO-rapport  
aan direct belanghebbenden is toegestaan.

Exemplaar nr. : 9  
Oplage : 30  
Aantal pagina's : 76 (incl. bijlagen,  
excl. RDP & distributielijst)  
Aantal bijlagen : 2

© 1997 TNO

TNO Fysisch en Elektronisch Laboratorium is onderdeel  
van de hoofdgroep TNO Defensieonderzoek  
waartoe verder behoren:

TNO Prins Maurits Laboratorium  
TNO Technische Menskunde



DTIC QUALITY INSPECTED

DISTRIBUTION STATEMENT A

Approved for public release;  
Distribution Unlimited

Nederlandse Organisatie voor toegepast-  
natuurwetenschappelijk onderzoek TNO

## Managementuittreksel

Titel : Software Specificaties BIAS  
Auteur(s) : Ir. H.G.M. Bennis, Ir. A.P.T.M. Onderwater  
Datum : september 1997  
Opdrachtnr. : A97KL790  
IWP-nr. : 7652  
Rapportnr. : FEL-97-A253

Bij de interceptie van draadloze overdrachtsystemen dienen deze, na analyse van inhoud en herkomst, geregistreerd te worden. Het wil nog al eens voorkomen dat onbekende overdrachtsystemen geïntercepteerd worden. De analyse van de structuur van deze onbekende overdrachtsystemen gebeurt tot op heden grotendeels met de hand. De KL voorziet dat de capaciteit voor de analyse van toekomstige onbekende overdrachtsystemen te kort schiet.

Daarom is de automatisering van dit analyseproces een voor de hand liggende oplossing. Aan deze behoefte zal voldaan worden door het *BlI Analyse Systeem (BIAS)*. Om een indruk te krijgen van hedendaagse analyse software, is na een *Request For Information (RFI)*, een inventarisatie uitgevoerd. De daaruit voortvloeiende informatie is gebruikt in het opstellen van de *Software Specificaties*. De *Software Specificaties* zijn beschreven volgens standaard MIL-STD-498 in de *Software Requirement Specifications (SRS)* evenals de *Recommendations Software Specifications (RSS)* welke het vervolg is op de SRS. De *Software Specificaties* kunnen worden opgedeeld in:

- Functionele Specificaties,
- Gebruikers Specificaties.

De voordelen van geautomatiseerde ondersteuning zijn meervoudig. Een analist zal:

- een beter inzicht krijgen in de data,
- sneller kunnen werken,
- beter onderbouwde analysebeslissingen kunnen nemen,
- meer en statistisch onderbouwde gereedschappen ter beschikking hebben.

Dit project is het tweede onderdeel uit een drieluik van opdrachten, bedoeld om te komen tot de implementatie van het *BlI Analyse Systeem* zoals dat beschreven is in de *Software Requirement Specifications (SRS)*. De implementatie van het *BlI Analyse Systeem* is het derde onderdeel.

## Samenvatting

Binnen de KL is, in verband met de werkzaamheden van een van zijn onderdelen, de behoefte gesteld voor een geautomatiseerd stuk gereedschap voor de analyse van digitale overdrachtsystemen. De analyse van de structuur van nog onbekende overdrachtsystemen gebeurt tot op heden nog grotendeels met de hand. De automatisering van dit analyseproces is een voor de hand liggende oplossing. Aan deze behoefte zal voldaan worden door het *Blt Analyse Systeem (BIAS)*. Een systeem waarmee een analist geautomatiseerd ondersteund wordt om:

- een beter inzicht te krijgen in de data,
- sneller te kunnen werken,
- beter onderbouwde analysebeslissingen te kunnen nemen,
- en meer en statistisch onderbouwde gereedschappen te hebben.

Voor de opzet van de *Software Specificaties* van *BIAS* is gebruik gemaakt van de resultaten van eerder door TNO-FEL uitgevoerd onderzoek en is gekeken naar de mogelijkheden die reeds bestaande analysesoftware biedt. De randvoorwaarden voor de functionaliteit van deze bestaande analysesoftware is geheel vrijgelaten. Om een overzicht van de beschikbare software te krijgen is een inventarisatie uitgevoerd aan de hand van een *Request For Information (RFI)*. Deze inventarisatie is niet opgenomen in dit document. De uit de inventarisatie voortvloeiende informatie is gebruikt bij het opstellen van de *Software Specificaties*. Deze *Software Specificaties* kunnen worden opgedeeld in:

- Functionele Specificaties,
- Gebruikers Specificaties.

De Software Specificaties zijn beschreven volgens standaard MIL-STD-498 in de *Software Requirement Specifications*, welke in bijlage A is opgenomen. De *Recommendations Software Specifications*, welke het vervolg is op de *Software Requirement Specifications*, is opgenomen in bijlage B. Hierin wordt een advies gegeven voor het derde deel van het drieluik, de implementatie.

## Inhoud

1.	Software Specificaties voor BIAS .....	5
2.	Aanbevelingen voor BIAS .....	6
3.	Ondertekening .....	7

### Bijlagen

A	Software Requirement Specification for BIAS
B	Recommendations Software Specifications for BIAS

## **1. Software Specificaties voor BIAS**

Dit hoofdstuk beschrijft de aanbevelingen voor de Software Specifications voor 'BIIt Analyse Systeem (BIAS)'. Aan de hand van deze aanbeveling wordt een beeld geschetst van de mogelijkheden om te komen tot een softwarepakket voor een analysesysteem ten behoeve van de analyse van digitale overdrachtsystemen. Deze specificaties zijn in bijlage A, " Software Requirement Specifications" gedocumenteerd volgens de MIL-STD-498.

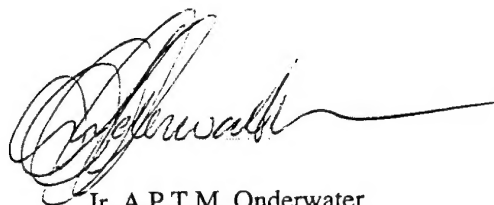
## **2. Aanbevelingen voor BIAS**

Naar aanleiding van het opstellen van de Software Requirement Specifications is er een document opgesteld waarin de Recommendations Software Specifications staan gerapporteerd. Deze rapportage is gegeven in bijlage B.

### 3. Ondertekening

A stylized, handwritten signature in black ink, consisting of several overlapping loops and a long horizontal stroke extending to the right.

Dr. ir. A.P.M. Zwamborn  
Groepsleider

A handwritten signature in black ink, featuring a series of loops and a long, sweeping horizontal line that extends to the right.

Ir. A.P.T.M. Onderwater  
Projectleider/Auteur

---

## **Bijlage A      Software Requirement Specification for BIAS**

De beschrijving van de SRS volgt op de volgende pagina's en kan beschouwd worden als een zelfstandig onderdeel binnen dit document.



# Software Requirements Specification for BIAS

Document identification 6026405.000/SRS001  
Contract no 861.3886.3963.11  
Edition 2.0  
Date 15 September 1997

Prepared by TNO Physics and Electronics Laboratory  
PO Box 96864  
2509 JG The Hague  
The Netherlands

Prepared for DMKL/INFO  
Telecom and Information Systems  
Van der Burghlaan 31  
2597 PC The Hague  
The Netherlands

## Approved by client

Function  
Name/initials  
Date

Function  
Name/initials  
Date

Function  
Name/initials  
Date

## Approved by TNO-FEL

Function  
Name/initials  
Date

Function  
Name/initials  
Date

Function  
Name/initials  
Date

The classification designation Ongerubriceerd is equivalent to Unclassified, Stg.  
Confidentieel is equivalent to Confidential and Stg. Geheim is equivalent to Secret.

All rights reserved. No part of this publication may be reproduced and/or published by print,  
photoprint, microfilm or any other means without the previous written consent of TNO.

## Configuration item history log

Edition	Date	Chapter	Description
0.1	220497	All	First draft
0.2	280597	All	Extensions
0.3	020697	All	MIL-STD 498 standard adopted
0.4	040797	All	Revisions accepted
1.0	090797	All	Final draft
1.01	140797	All	Minor corrections
1.02	210797	All	Consistency check
1.03	220797	All	Revisions accepted
1.04	120897	All	Internal review accepted
1.05	140897	All	Review by client and DCA-KL accepted
2.0	210897	All	Final Version

## Contents

1.	Scope .....	1
1.1	Identification .....	1
1.2	System overview .....	1
1.3	Document overview .....	2
2.	Referenced documents .....	3
3.	Requirements .....	4
3.1	Required states and modes .....	4
3.2	CSCI capability requirements (BBC_) .....	4
3.2.1	General Requirements (BGR_) .....	5
3.2.2	File handling (FIH_) .....	7
3.2.3	Editing (EDI_) .....	8
3.2.4	Display of data (PRD_) .....	10
3.2.5	Mathematical Functions (MSF_) .....	12
3.2.6	Bitstream Manipulation (BIM_) .....	17
3.2.7	Tools (TOO_) .....	23
3.2.8	Window Functions (WIN_) .....	44
3.2.9	Help (HEL_) .....	45
3.3	CSCI external interface requirements (EIR_) .....	45
3.3.1	Interface identification and diagrams .....	45
3.3.2	Interface with Receiver/Demodulator .....	45
3.3.3	Interface with existing Data Handling System .....	45
3.3.4	Interface with External Resources and Storage Media .....	46
3.3.5	Interface with Printers .....	46
3.4	CSCI internal interface requirements .....	46
3.5	CSCI internal data requirements .....	46
3.5.1	Requirements (DAT_) .....	46
3.5.2	Constraints .....	47
3.6	Adaptation requirements .....	47
3.6.1	Installation-dependent data .....	47
3.6.2	Operational parameters .....	47
3.7	Safety requirements .....	47
3.8	Security and privacy requirements .....	48
3.9	CSCI environment requirements .....	48
3.10	Computer resource requirements .....	48
3.10.1	Computer hardware requirements .....	48
3.10.2	Computer hardware resource utilisation requirements .....	48
3.10.3	Computer software requirements .....	48
3.10.4	Computer communications requirements .....	48
3.11	Software quality factors .....	48
3.12	Design and implementation constraints .....	48
3.13	Personnel-related requirements (HER_) .....	48

3.14	Training-related requirements.....	49
3.15	Logistics-related requirements.....	49
3.16	Other requirements .....	49
3.17	Packaging requirements .....	49
3.18	Precedence and criticality of requirements .....	49
4.	Qualification provisions .....	50
4.1	Qualification methods .....	50
4.2	Special qualification requirements .....	50
5.	Requirements traceability.....	51
6.	Notes.....	52
6.1	Abbreviations .....	52
6.2	Definitions .....	52
6.3	Items to be defined.....	52
6.4	General information .....	52
	Appendix	
A	Description Language .....	A-1

## Figures and tables

### Figures

Figure 1.1: System diagram .....	1
Figure 3.1: Presentation of data in windows .....	6
Figure 3.2: Example of a periodicity analysis output .....	26
Figure 3.3: Schematic overview of linear feedback shift-register .....	27
Figure 3.4: Example of pattern statistics analysis. ....	29
Figure 3.5: Example of a Convolution Encoder .....	32
Figure 3.6: Scematic view of a concatenated Coder.....	33
Figure 3.7: Example of a received Viterbi Code .....	36
Figure 3.8: Model of a linear feedback shift-register .....	38
Figure 3.9: Example of a convolution encoder .....	39
Figure 3.10: Concatenated Code Schema. ....	40
Figure 3.11: Example of a received Viterbi Code .....	42

### Tables

Table 3.1: Main Menu .....	5
Table 3.2: File menu .....	7
Table 3.3: Edit menu .....	8
Table 3.4: Display menu .....	10
Table 3.5: Mathematics menu .....	12
Table 3.6: Bitstream menu .....	17
Table 3.7: Tools menu .....	23
Table 3.8: Verification Functions .....	24
Table 3.9: Cipher Analysis .....	25
Table 3.10: Pattern Statistics .....	28
Table 3.11: Synthesis Functions .....	37
Table 3.12: Macros .....	44
Table 3.13: Window menu .....	44
Table 3.14: Help menu .....	45

## 1. Scope

### 1.1 Identification

This document specifies the requirements for a BIt Analysis System(BIAS). The objective of BIAS is to provide the user with means to analyse the structure of complex data communication systems.

Input to BIAS shall be the real time output (both analogue and digital) of demodulation systems, or pre-recorded analogue and digital signals.

The analysis of communication systems will be performed off-line.

The purpose of this document is twofold:

- list all requirements in detail as unambiguous, independent and testable items
- provide a checklist for examining solutions offered by vendors.

### 1.2 System overview

#### Goal

The goal of BIAS is to serve partly as a supporting, partly as an executing system, in the analysis of discrete transmission systems by an analyst.

#### System diagram

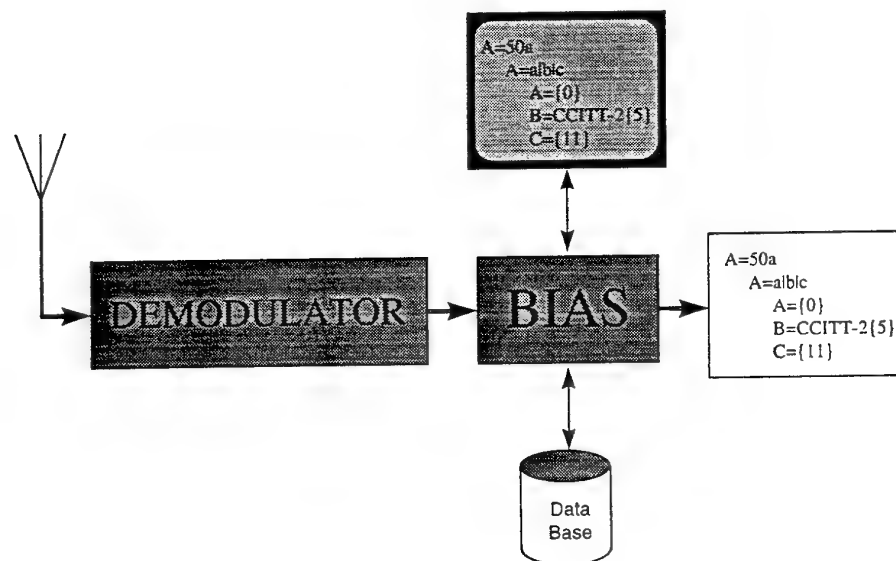


Figure 1.1: System diagram

**External interfaces**  
(Not Applicable) NA

### **1.3 Document overview**

NA

## 2. Referenced documents

In the text is referred to:

Document identification	Document name	Edition	Date	Published by
[1]	Analyse van digitale overdrachtsystemen	-	1996	TNO-FEL
[2]	Chi square test	-	1996	The Pennsylvania State University, 1996
[3]	Shift-Register Synthesis and BCH Decoding	Vol 1	1969	IEEE Transactions on Information Theory
[4]	Basic Concepts in Information Theory and Coding	-	1994	Plenum Press
[5]	Communication Systems	Third Edition	1986	Mc-Graw-Hill
[6]	The Theory of Error Correcting Codes	-	1977	North-Holland Pub.Co



### 3. Requirements

#### 3.1 Required states and modes

NA

#### 3.2 CSCI capability requirements (BBC\_)

BBC\_010: Basic BIAS capabilities shall provide for:

1. file handling (including printing),
2. editing,
3. presentation of data,
4. performing mathematical, logical and statistical functions,
5. bitstream manipulation,
6. using tools (verification functions, synthesis functions, macros, code generators, coding tables)
7. window management,
8. help.

BBC\_020: Functions shall be selectable using a Pull Down Menu structure.

BBC\_030: Standard Windows Shortcut keys shall be implemented.

BBC\_040: It shall be possible to define user shortcuts.

BBC\_050: Menu items may result in direct action or require further specification using one or more dialogue boxes as necessary.

BBC\_060: Toolbars with direct action buttons shall be provided. These toolbars shall be customisable.

BBC\_065: It shall be possible to define, edit and store user profiles, in which at least the following items can be specified:

- Tool bars in use
- Layout of tool bars

BBC\_070: The basic capabilities shall be augmented by allowing linking into BIAS of other programs such as a spreadsheet program and an advanced mathematics program (e.g. Excel and MatLab). Exchange of data between BIAS and the linked programs, and between the linked programs, shall be possible.

BBC\_080: The system shall be portable to at least the following platforms:

- TBD

Only for the specification of requirements the menu structure as given in Table 3.1 to Table 3.14 is used. Other implementations may be considered.  
Specifications are detailed in the sections 3.2.1 to 3.2.9.

Table 3.1: Main Menu

File	Edit	Display	Math	Bitstream	Tools	Window		Help
------	------	---------	------	-----------	-------	--------	--	------

### 3.2.1 General Requirements (BGR\_)

- BGR\_010: The program shall be structured as a shell for manipulating and editing windows, with all application functionality being add-ins.
- BGR\_020: The functionality described in this Requirement Specification shall be delivered by the vendor. It shall however be possible for the user or any third party to add extra functionality.
- BGR\_030: The menu structure shall allow for customising by the user, following the addition of extra functionality.
- BGR\_035: Applicable parameters for functions are defined as the areas selected in the presentations using the selection facility.
- BGR\_040: Macros are defined as combinations of the basic BIAS capabilities BBC\_010, items 1 and 3..7 (see section 3.2).
- BGR\_041: A set of predefined macros shall be delivered with the system.
- BGR\_050: It shall be possible to define (record) custom macros.
- BGR\_060: Analyser functions resulting in a new presentation of the data shall be logged with the results.
- BGR\_070: At request of the operator, all functions sequentially performed to obtain a selected presentation shall be gathered and translated into a description language for the data structure.

#### 3.2.1.1 Auto Saving of Current Status

- BGR\_080: The system shall automatically save files  $n$  minutes after the first change of each file.
- BGR\_090: The number of minutes is to be specified by the user.

#### 3.2.1.2 Status Recall after Irregular Shutdown

- BGR\_100: After an irregular shutdown the system shall automatically recover the last autosaved status at start up.

#### 3.2.1.3 Multiple Window Presentation

The following requirements are applicable:

- BGR\_110: There may be no restriction on the number of windows in use at any moment
- BGR\_120: The usual facilities for manipulating windows
- closing
  - arranging
  - iconising
  - moving
  - resizing
  - on top (Y/N)

shall be available

#### 3.2.1.4 Recursiveness of Operations

BGR\_130: All operations performed on data in one or more windows shall result in the presentation of the results in a new window.

BGR\_140: A hierarchy of windows shall be maintained.

BGR\_150: A view of the hierarchy shall be provided.

BGR\_160: It shall be possible to select and open windows in this view.

Figure 3.1 illustrates this requirement.

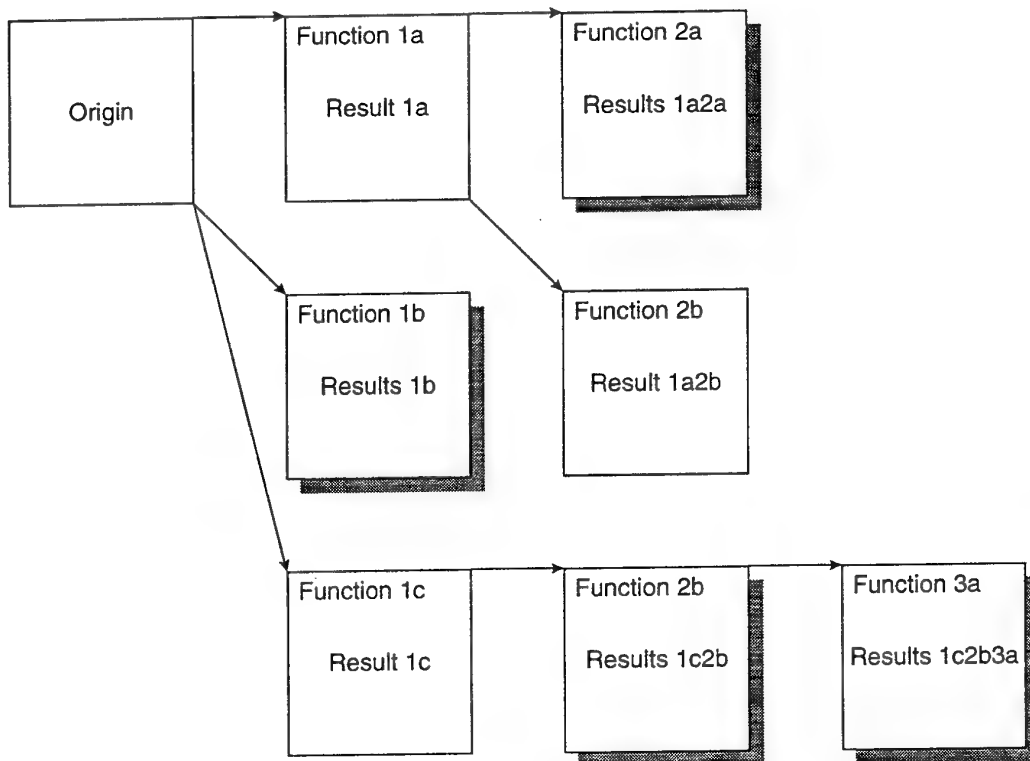


Figure 3.1: Presentation of data in windows

### 3.2.2 File handling (FIH\_)

Table 3.2: File menu

File	Shortcut
New ...	Ctrl+N
Open ...	Ctrl+O
Close	
Save	Ctrl+S
Save As ...	
Save All	
Print ...	Ctrl+P
Print Preview	
Page Set-up ...	
Print Set-up ...	
Preferences ...	
1 Filename-a	
2 Filename-b	
Exit	

FIH\_010: The following standard facilities for file handling and printing shall be available.

- New
- Open
- Close
- Save
- Save As ...
- Save All
- Print
- Print Preview
- Page Set-up ...
- Print Set-up ...
- Exit

FIH\_015: The use of long file names shall be supported.

FIH\_020: There shall be a list of recently used files for direct selection.

#### 3.2.2.1 Preferences ...

FIH\_030: It shall be possible to maintain and edit a list of preferred (default) settings. This list shall at least include:

- Autosave settings (BGR\_090)
- User profiles

### 3.2.3 Editing (EDI\_)

Table 3.3: Edit menu

Edit	(Shortcut)
Undo	Ctrl+Z
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Clear	Del
Select All	Ctrl+A
Find ...	Ctrl+F
Replace ...	Ctrl+H
Go To ...	Ctrl+G
Bookmark ...	

All data shall be displayed in windows for which edit facilities are available  
These facilities shall include

- all standard facilities that may be expected from a modern editor, and
- facilities for the support of BIAS functions.

The edit facilities are required for two purposes:

1. general editing in windows
2. selecting parameters for operations

EDI\_020: The general edit functions

- insertion
- deletion
- line length variation
- copy functionality
- paste functionality
- inter window copy & paste

shall not create a new view of the data in a new window.

EDI\_030: Functions performed outside the editor shall always create a new window for the display of the results.

#### 3.2.3.1 General Edit Requirements

The following general requirements apply:

EDI\_040: File sizes of at least 1 M DataElements shall be possible

EDI\_050: Line lengths of at least 2 K DataElements shall be possible

EDI\_060: Edit functions to be provided shall include:

- Undo
- Cut
- Copy
- Paste
- Clear

- Select All
- Find ...
- Replace ...
- Go To ...
- Bookmark ...

### 3.2.3.2 Facilities for the support of BIAS functions

#### 3.2.3.2.1 Selection of DataElements

DataElements must be selectable in the following ways:

- EDI\_070: any number of adjacent DataElements
- selection starts at insertion point, possibly extending past end of line
  - selection can be made in forward or backward direction
- EDI\_080: any number of DataElements in a column, from insertion point
- selection starts at insertion point
  - selection can be made in upward or downward direction
- EDI\_090: block select
- a block is defined as a rectangular portion of DataElements
  - a block shall not extend beyond the line break of any row in the selection (left aligned)
  - a block shall not extend beyond the start of line of any row in the selection (right aligned)
- EDI\_100: column select
- column select shall extend in upward direction to the first line containing a DataElement in the selected column, and in downward direction to the last line containing a DataElement in the selected column
- EDI\_110: selection of disjunct regions of DataElements
- any combination of the selections EDI\_070 .. EDI\_100 above must be possible
- EDI\_120: For each function to be performed it shall be possible to specify the way the selection is to be used as a (set of) parameter(s).

#### 3.2.3.2.2 Insert DataElements

- EDI\_130: It shall be possible to insert at the insertion point a DataElement selectable from a symbol table.

#### 3.2.3.2.3 Find ...

- EDI\_140: It shall be possible to find any sequence of DataElements.
- EDI\_150: Wildcards must be allowed in the specified sequence.
- EDI\_160: The action shall be repeatable.

#### 3.2.3.2.4 Replace ...

- EDI\_170: It shall be possible to replace a found sequence with any other predefined sequence.
- EDI\_180: Replacement shall have to be confirmed.

- EDI\_190: The action shall be repeatable.  
 EDI\_200: Unconditional replacement shall be optional.

#### 3.2.3.2.5 Go To ...

- EDI\_210: It shall be possible to go to a specified location in the file.  
 The location shall be either:
- a specified DataElement count
  - a selected bookmark.

#### 3.2.3.2.6 Bookmark ...

- EDI\_220: It shall be possible to place a named bookmark at any selection of DataSymbols.

### 3.2.4 Display of data (PRD\_)

Table 3.4: Display menu

Display	(Shortcut)
• Normal	
Analysis Structure	
Full Screen	
Description Language	
Font ...	
Highlight ...	
Line Length ...	
Insert Line Break	
Split Line	
Right Align (c.q. Left Align)	
✓ Toolbars ...	
✓ Ruler	
✓ Status Bar	

Essential to the usefulness of the system is the flexibility of how to display data to the user, both onscreen and in hardcopy, and the ability for the operator to manipulate data. Therefore, the display facilities must allow the operator to view the data in a way most suitable to his needs.

#### 3.2.4.1 General Requirements

The display of data shall provide for the following facilities:

- PRD\_020: The number of DataElements before insertion point/cursor/current screen shall be displayed  
 PRD\_030: There shall be a ruler indicating the position of DataElements on a line

#### 3.2.4.2 Analysis Structure

- PRD\_040: It shall be possible to select a graphical representation of the analysis structure.  
 PRD\_050: The hierarchy of all files must be shown.

- PRD\_060: The view of the analysis structure shall allow the selection of a file for viewing.
- PRD\_070: It shall be possible to retrieve a summary of a selected file.

#### 3.2.4.3 Description Language

- PRD\_080: There shall be a language for the description of the structure of a signal under analysis.
- PRD\_090: The language shall support the description of all constructions for which analysis functions are available.

Appendix A contains a possible implementation of this language.

#### 3.2.4.4 Font

Font types and sizes shall allow for:

- PRD\_100: a global representation of large amounts of data
- PRD\_110: a detailed representation of a local view
- PRD\_120: intermediate representations as feasible

#### Font selection

- PRD\_130: Font selection shall apply for the entire file being displayed.
- PRD\_140: There shall be an indication of the relative size of each font.

#### Font Editing

- PRD\_150: Font Bitmaps shall be editable by the user.

#### 3.2.4.5 Highlighting of selections

- PRD\_160: It must be possible to highlight any selection in at least the following ways:
- bolding
  - underlining
  - selectable colour
    - colour selectable from palette
    - there must be a default palette
    - there shall be a user definable palette

#### 3.2.4.6 Line Length ...

- PRD\_170: It shall be possible for the operator to vary the number of DataElements on a line. The following means shall be available for this purpose:
- enter the linelength using the keyboard
  - increase/decrease the linelength by 1 DataElement
  - increase/decrease the linelength by 5 DataElements

#### 3.2.4.7 Insert Line Break

- PRD\_180: It shall be possible to insert line breaks at selectable points.
- PRD\_190: It shall be possible to insert line breaks before or after each found search pattern.
- PRD\_200: A line break alters the linelength of the current line only.
- PRD\_210: It shall be possible to delete any previously inserted line break.



### 3.2.4.8 Split Line

PRD\_220: Splitting a line is analogous to inserting a line break. The difference is that the next line will only contain the rest of the original line.

### 3.2.4.9 Left Align / Right Align

PRD\_230: When linebreaks and/or line splits have been inserted, and thus lines of different lengths exist, it shall be possible to select left or right alignment of lines.

### 3.2.4.10 Toolbars, Ruler and Status Bar

PRD\_240: It shall be possible to create toolbars with direct action buttons for frequently used operations.

PRD\_241: It shall be possible to create and edit user toolbar buttons

PRD\_250: The display of each toolbar shall be selectable.

## 3.2.5 Mathematical Functions (MSF\_)

Table 3.5: Mathematics menu

Mathematics	(Shortcut)
2 Modulo ...	
Absolute Value	
Add	
Summation $\Sigma$	
Average	
Standard Deviation	
Subtract	
Multiply	
Product $\Pi$	
Dot Product	
Vector Product	
Divide	
2 Exponentiation ...	
2 Root Extraction ...	
10 Logarithm ...	

These functions are mainly included to support the BIAS analytical functions.

As applicable mathematical operations have to be performed in  $GF(\mathbb{K}^+)$  for integer variables, and in  $\{\mathbb{R}\}$  for non integer variables.

### 3.2.5.1 Modulo

MSF\_010:

#### Purpose

Perform Mathematical operations Modulo  $n$ .

#### Input Parameters

- $n$  from Keyboard | Menu

$$2 \leq \text{DataCode Value} \leq n \quad (3.1)$$

**Output Parameters**

- An unsigned DataElement

**3.2.5.2 Absolute Value**

MSF\_020:

**Purpose**

Determine the Absolute Value of a Parameter

$$B = |A| \quad (3.2)$$

**Input Parameters**

- Value  $A$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $B$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $B$  in the notation for DataType  $\Omega$

**3.2.5.3 Add**

MSF\_030:

**Purpose**

Addition of two parameters.

$$C = A + B \quad (3.3)$$

**Input Parameters**

- Value  $A$  in the notation for DataType  $\Psi$
- Value  $B$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $C$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $C$  in the notation for DataType  $\Omega$

**3.2.5.4 Summation  $\Sigma$** 

MSF\_040:

**Purpose**

Calculate the Sum of a range of parameters

$$B = \sum_{n=x}^{n=y} A_n \quad (3.4)$$

**Input Parameters**

- Values  $A_n$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $B$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $B$  in the notation for DataType  $\Omega$

**3.2.5.5 Average**

MSF\_050:

**Purpose**

Calculate the average value of a range of parameters

$$B = \frac{\sum_{n=x}^{n=y} A_n}{y - (x - 1)} \quad (3.5)$$

**Input Parameters**

- Values  $A_n$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $B$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $B$  in the notation for DataType  $\Omega$

### 3.2.5.6 Standard Deviation

MSF\_060:

**Purpose**

Calculate the standard deviation for a range of parameters

$$B = \sqrt{\frac{\sum_{n=x}^{n=y} \left( A_n - \frac{\sum_{m=x}^{m=y} A_m}{y - (x - 1)} \right)^2}{y - (x - 1)}} \quad (3.6)$$

**Input Parameters**

- Values  $A_n$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $B$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $B$  in the notation for DataType  $\Omega$

### 3.2.5.7 Subtract

MSF\_070:

**Purpose**

Subtraction of two parameters

$$C = A - B \quad (3.7)$$

**Input Parameters**

- Value  $A$  in the notation for DataType  $\Psi$
- Value  $B$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $C$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $C$  in the notation for DataType  $\Omega$

### 3.2.5.8 Multiply

MSF\_080:

**Purpose**

Multiplication of two parameters

$$C = A \times B \quad (3.8)$$

**Input Parameters**

- Value  $A$  in the notation for DataType  $\Psi$
- Value  $B$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $C$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $C$  in the notation for DataType  $\Omega$

**3.2.5.9 Product  $\Pi$**

MSF\_090:

**Purpose**

Calculate the Product of range of parameters

$$B = \prod_{n=x}^{n=y} A_n \quad (3.9)$$

**Input Parameters**

- Values  $A_n$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $B$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $B$  in the notation for DataType  $\Omega$

**3.2.5.10 Dot-product**

MSF\_100:

**Purpose**

Calculate the Dot-product of 2 equal length vectors

$$C = \overline{A} \cdot \overline{B} \quad (3.10)$$

**Input Parameters**

- Vector  $A$  in the notation for DataType  $\Psi$
- Vector  $B$  in the notation for DataType  $\Psi$

**Output Parameters**

- Value  $C$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $C$  in the notation for DataType  $\Omega$

**3.2.5.11 Vector-product**

MSF\_110:

**Purpose**

Calculate the Vector-product of 2 equal length sequences of DataElements

$$\overline{C} = \overline{A} \times \overline{B} \quad (3.11)$$

**Input Parameters**

- Vector  $A$  in the notation for DataType  $\Psi$
- Vector  $B$  in the notation for DataType  $\Psi$

**Output Parameters**

- Vector  $C$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $C$  in the notation for DataType  $\Omega$

#### 3.2.5.12 Divide

MSF\_120:

##### Purpose

Division of two parameters

$$C = A / B \quad (3.12)$$

##### Input Parameters

- Value  $A$  in the notation for DataType  $\Psi$
- Value  $B$  in the notation for DataType  $\Psi$

##### Output Parameters

- Value  $C$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $C$  in the notation for DataType  $\Omega$

#### 3.2.5.13 Exponentiation

MSF\_130:

##### Purpose

Raise a parameter to the power  $n$

$$B = A^n \quad (3.13)$$

##### Input Parameters

- Value  $A_n$  in the notation for DataType  $\Psi$
- Value  $n$  from Keyboard, in the notation for DataType  $\mathbb{R}$ .

##### Output Parameters

- Value  $B$  in the notation for DataType  $\Psi$  if  $\Psi$  is a ring,  
else value  $B$  in the notation for DataType  $\Omega$

#### 3.2.5.14 Root Extraction

MSF\_140:

##### Purpose

Calculate the root (power  $n$ ) of a parameter

$$B = \sqrt[n]{A} \quad (3.14)$$

##### Input Parameters

- Value  $A$  in the notation for DataType  $\Psi$
- Value  $n$  from Keyboard, in the notation for DataType  $\mathbb{N}$

##### Output Parameters

- Value  $B$  in the notation for DataType  $\Omega$

#### 3.2.5.15 Logarithm

MSF\_150:

##### Purpose

Calculate the Logarithm (power  $n$ ) of a parameter

$$B = n \log A \quad (3.15)$$

#### Input Parameters

- Value  $A$  in the notation for DataType  $\Psi$
- Value  $n$  from Keyboard, in the notation for DataType  $\aleph$

#### Output Parameters

- Value  $B$  in the notation for DataType  $\Omega$

### 3.2.6 Bitstream Manipulation (BIM\_)

Table 3.6: Bitstream menu

Bitstream	(Shortcut)
Undo	
Mask/Unmask	
Hide/Unhide Mask	
Copy	
Cut	
Paste	
Periodic Delete	
Periodic Insert	
INVERT	
AND	
OR	
XOR	
NAND	
NOR	
XNOR	
Rotate ...	
Shift ...	
Mirror ...	
Multiplex ...	
De-Multiplex ...	
Interleave ...	
De-Interleave ...	

Where applicable functions are specified using the Description Language as specified in Appendix A.

#### 3.2.6.1 Undo

BIM\_005: There shall be an Undo facility for at least the last 10 actions performed by the operator.

#### 3.2.6.2 Mask/Unmask

BIM\_010: It shall be possible to define any selection of DataElements as Masked.

BIM\_020: It shall be possible to define any selection of masked DataElements as Unmasked.

BIM\_030: Masked DataElements shall be recognisable.

### 3.2.6.3 Hide/Unhide Mask

BIM\_040: It shall be possible to Hide / Unhide masked DataElements.

BIM\_050: Hidden DataElements shall (optionally) leave open spaces in the display of DataElements.

### 3.2.6.4 Copy

BIM\_060: It shall be possible to copy any selection of DataElements to a clipboard.

### 3.2.6.5 Cut

BIM\_070: It shall be possible to cut any selection of DataElements to a clipboard.

### 3.2.6.6 Paste

BIM\_080: It shall be possible to paste the contents of the clipboard at the insertion point.

### 3.2.6.7 Periodic Delete

BIM\_090: It shall be possible to delete a selected sequence of DataElements in any linear periodic way.

### 3.2.6.8 Periodic Insert

BIM\_100: It shall be possible to insert a selected sequence of DataElements in any linear periodic way.

### 3.2.6.9 Invert

BIM\_110:

#### Purpose

The purpose of INVERTing is to get the mathematically opposite value of a selected value. If a selection is made of a sequence of DataElements, than the DataElements will be piece wise inverted.

$$B = INVERT(A) \quad (3.16)$$

#### Input Parameters

- A sequence of DataElements,  $A$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

#### Output Parameters

- A sequence of the same number of DataElements,  $B$ , in the notation for DataType  $\Psi$

### 3.2.6.10 AND

BIM\_120:

#### Purpose

The purpose of ANDing is to get the logical AND value of two selected equal length sequences. If a selection is made of two sequences of DataElements, than the DataElements will be piece wise calculated.

$$Z = AND(X, Y) \quad (3.17)$$

**Input Parameters**

- A sequence of DataElements,  $X$ , of at least 1 DataElement, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $Y$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of the same number of DataElements,  $Z$ , in the notation for DataType  $\Psi$

**3.2.6.11 OR**

BIM\_130:

**Purpose**

The purpose of ORing is to get the logical OR of a selected value. If a selection is made of a sequence of DataElements the DataElements will be piece wise calculated.

$$Z = OR(X, Y) \quad (3.18)$$

**Input Parameters**

- A sequence of DataElements,  $X$ , of at least 1 DataElement in the notation for DataType  $\Psi$
- A sequence of DataElements,  $Y$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of the same number of DataElements,  $Z$ , in the notation for DataType  $\Psi$

**3.2.6.12 XOR**

BIM\_140:

**Purpose**

The purpose of XORing is to get the logical XOR of two selected equal length sequences. If a selection is made of a sequence of DataElements the DataElements will be piece wise calculated.

$$Z = XOR(X, Y) \quad (3.19)$$

**Input Parameters**

- A sequence of DataElements,  $X$ , of at least 1 DataElement, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $Y$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of the same number of DataElements,  $Z$ , in the notation for DataType  $\Psi$

**3.2.6.13 NAND**

BIM\_150:

**Purpose**

The purpose of NANDing is to get the logical NAND of two selected equal length sequences. If a selection is made of a sequence of DataElements the DataElements will be piece wise calculated.



$$Z = NAND (X, Y) \quad (3.20)$$

**Input Parameters**

- A sequence of DataElements,  $X$ , of at least 1 DataElement, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $Y$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of the same number of DataElements,  $Z$  in the notation for DataType  $\Psi$

**3.2.6.14 NOR**

BIM\_160:

**Purpose**

The purpose of NORing is to get the logical NOR of two selected equal length sequences. If a selection is made of a sequence of DataElements the DataElements will be piece wise calculated.

$$Z = NOR (X, Y) \quad (3.21)$$

**Input Parameters**

- A sequence of DataElements,  $X$ , of at least 1 DataElement, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $Y$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of the same number of DataElements,  $Z$ , in the notation for DataType  $\Psi$

**3.2.6.15 NXOR**

BIM\_170:

**Purpose**

The purpose of NXORing is to get the logical NXOR of two selected equal length sequences. If a selection is made of a sequence of DataElements the DataElements will be piece wise calculated.

$$Z = NXOR (X, Y) \quad (3.22)$$

**Input Parameters**

- A sequence of DataElements,  $x$ , of at least 1 DataElement, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $y$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of the same number of DataElements,  $Z$ , in the notation for DataType  $\Psi$

**3.2.6.16 Rotate**

BIM\_180:

**Purpose**

Obtain a cyclically rotated version of a selected sequence of DataElements.

$B = \text{ROTATE } a \langle n, \text{dir} \rangle$  (3.23)

**Input Parameters**

- Sequence,  $a$ , in the notation for DataType  $\Psi$
- $n, \text{dir}$ : from keyboard (dir is direction)

**Output Parameters**

- Sequence,  $B$ , in the notation for DataType  $\Psi$

**3.2.6.17 Shift**

BIM\_190:

**Purpose**

Obtain a shifted version of a selected sequence of DataElements.

$B = \text{SHIFT } a \langle n, \text{dir}, \text{fill} \rangle$  (3.24)

$n, \text{dir}$ : from keyboard (dir is direction)

$\text{fill}$ : selected from symbol table through the keyboard

**Input Parameters**

- Sequence,  $a$ , in the notation for DataType  $\Psi$
- $n, \text{dir}, \text{fill}$ : from keyboard

**Output Parameters**

- Sequence in the notation for DataType  $\Psi$

**3.2.6.18 Mirror**

BIM\_200:

**Purpose**

Obtain a mirrored version of a selected sequence of DataElements.

$B = \text{MIRROR } a[n]$  (3.25)

**Input Parameters**

- Sequence,  $a$ , in the notation for DataType  $\Psi$
- $n$ : from keyboard ( $n$  is position in sequence)

**Output Parameters**

- Sequence,  $B$ , in the notation for DataType  $\Psi$

**3.2.6.19 Multiplex**

BIM\_210:

**Purpose**

Obtain the multiplex of  $n$  sequences of DataElements

$A = \text{MUX } b[n] \langle i \rangle$  (3.26)

$A = \text{MUX } (x, y, \dots, z) \langle i \rangle$

**Input Parameters**

- Sequences,  $b$ , possibly of different lengths, in the notation for DataTypes  $\Psi_1.. \Psi_n$
- $i, n$ : from keyboard

**Output Parameters**

- Sequence,  $A$ , in the notation for DataType  $\Psi$

**3.2.6.20 De-multiplex**

BIM\_220:

**Purpose**

Obtain n demultiplexed sequences of DataElements from a sequence of DataElements

$$A[n] = \text{DEMUX } b \langle i \rangle \quad (3.27)$$

$$A(x, y, \dots, z) = \text{DEMUX } \langle i \rangle$$

**Input Parameters**

- Sequence,  $b$ , in the notation for DataType  $\Psi$
- $i, n$ : from keyboard

**Output Parameters**

- Sequences,  $A$ , possibly of different lengths, in the notation for DataTypes  $\Psi_1.. \Psi_n$

**3.2.6.21 Interleave**

BIM\_230:

**Purpose**

Obtain an interleaved version of a selected sequence of DataElements.

$$(A) = \text{INTERLEAVE } b \langle n_1, n_2 \rangle \quad (3.28)$$

**Input Parameters**

- Sequence,  $b$ , in the notation for DataType  $\Psi$
- $n_1, n_2$ : from keyboard

**Output Parameters**

- Sequence,  $A$ , in the notation for DataType  $\Psi$

**3.2.6.22 De-interleave**

BIM\_240:

**Purpose**

Obtain an interleaved version of a selected sequence of DataElements.

$$(A) = \text{DEINTERLEAVE } b \langle n_1, n_2 \rangle \quad (3.29)$$

**Input Parameters**

- Sequence,  $b$ , in the notation for DataType  $\Psi$
- $n_1, n_2$ : from keyboard

**Output Parameters**

- Sequence,  $A$ , in the notation for DataType  $\Psi$

### 3.2.7 Tools (TOO\_)

Table 3.7: Tools menu

Tools	(Shortcut)
Verification Functions ...	
Maintain Verification Functions ...	
Synthesis Functions ...	
Maintain Synthesis Functions ...	
Macros ...	
Maintain macros ...	

This chapter is divided into three major parts;

1. functions used for verification
2. functions used to synthesize a sequence of DataElements according to a specific function.
3. macro functions

The first part considers functions that will give an answer to certain analytical questions, which might be a polynomial, a number or an image.

In the second part, the synthesis functions are meant to generate a sequence of DataElements which might be used to verify or correct a received sequence of DataElements in the original stream. This feature will mostly be used for error correcting codes.

The third part incorporates sets of functions which are normally executed by the analyst and which are now executed by BIAS without analyst intervention, in a batch file kind of style.

#### 3.2.7.1 Verification Functions ...

This paragraph lists functions that shall be available to an analyst to get answers about the existence of patterns, sequences of DataElements, code schemes, etc. Answers may be visualised by means of polynomials, sequences of DataElements or images. An analyst shall be supported in his work by these verification functions.

TOO\_005: The results of the functions shall be presented in a result list with a confidence level per reported possible result.

Table 3.8: Verification Functions

Verification Functions	Shortcut
Auto Correlation	
Cross Correlation	
Discrete Fourier Analysis	
Cipher Analysis	
Pattern Statistics	
Coding Scheme Analysis	
Hamming Weight Analysis	
Hamming Distance Analysis	
CRC Analysis	
BCH Analysis	
Viterbi Analysis	
Stuffing Analysis	
Tree Code Analysis	

### 3.2.7.1.1 Auto Correlation

TOO\_010:

#### Purpose

The function *Auto Correlation* can be performed on a sequence of DataElements only. The sequence should consist of at least 2 DataElements, which need to represent a value, not a label. The result is a measure for the level to which this sequence of DataElements equals a shifted version of itself. The comparison is done over all the DataElements in the sequence of DataElements in a circular manner, i.e. the correlation will connect the end of the sequence with its beginning.

$$\psi_{xx}(\tau) = \frac{1}{N} \sum_{i=1}^N x_i x_j, \quad j = i - \tau \quad (3.30)$$

#### Input Parameters

- A sequence of DataElements,  $x$ , of at least 2 DataElements, in the notation for DataType  $\Psi$

#### Output Parameters

- A number representing the level of similarity, over a certain displacement,  $\tau$ , in this sequence of DataElements  $x$ . The result may also be a graph or a sequence of numbers, representing the level of correlation for  $\tau$  ranging the total length of the sequence

### 3.2.7.1.2 Cross Correlation

TOO\_020:

#### Purpose

The function *Cross Correlation* can be performed on two different sequences of DataElements only. The sequence of DataElements should consist of at least one DataElements, which need to represent a value, not a label. The result is a measure for the level to which these sequences of DataElements match. The comparison is done over all the DataElements in the sequence of DataElements in a circular manner.

$$\psi_{xy}(\tau) = \frac{1}{N} \sum_{i=1}^N x_i y_{i-\tau}, \quad j = i - \tau \quad (3.31)$$

**Input Parameters**

- A sequence of DataElements,  $x$ , of at least 1 DataElement, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $y$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

**Output Parameters**

- A number representing the level of similarity between the two sequences of DataElements, over a certain displacement,  $\tau$ . The result may also be a graph or a sequence of numbers, representing the level of correlation for  $\tau$  ranging the total length of the sequence

**3.2.7.1.3 Discrete Fourier Analysis**

TOO\_030:

**Purpose**

The function handles the entered sequence of DataElements as a sampled sequence of DataElements from the time domain. The result is a the representation of this signal in frequency domain.

$$S(m) = \sum_{i=0}^{N-1} x_i w_N^{im}, \quad m = 0, 1, \dots, N-1; w_N = e^{-j\frac{2\pi}{N}} \quad (3.32)$$

**Input Parameters**

- A sequence of DataElements,  $x$ , of at least 2 DataElements, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of DataElements,  $S$ , in the notation for DataType  $\mathfrak{S}$

**3.2.7.1.4 Cipher Analysis ...**

Table 3.9: Cipher Analysis

Cipher Analysis	Shortcut
Compression Recognition	
Periodicity Analysis	
Chi Square test	
Massey Berlekamp Algorithm	
Polynomial Division	

The need for cipher analysis grows as industry sells commercial of the shelf transmitters which support data encryption on an ever larger scale. Before starting with the analysis of sequence of DataElements it is important to check whether this sequence of DataElements is encrypted. Depending on the type of encryption, decryption becomes harder and harder and the effort must be put into the breaking of these encryptions becomes more and more futile. For the purpose of Cipher analysis the following functions shall be provided:

#### TOO\_040: Compression Recognition

##### Description

The tool is able to show whether or not a sequence of DataElements is compressed.

Characteristics of compression are:

- relatively short DataElement bursts
- range of burst length versus number of bursts close to or equal to that of cipher sequences
- no periodicity

##### Input Parameters

- A sequence of DataElements,  $x$ , of at least 2 DataElements, in the notation for DataType  $\Psi$

##### Output Parameters

- compression ratio
- possible compression method

#### TOO\_050: Periodicity Analysis

##### Description

The tool is able to show whether or not a sequence of DataElements contains any form of periodicity. The result is a graph which shows a number of peaks. The analyst shall be supported by additional built-in tools for the interpretation of the graph.

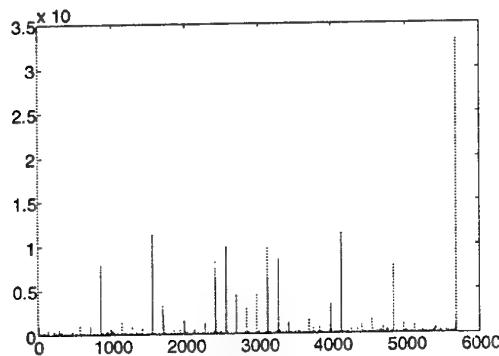


Figure 3.2: Example of a periodicity analysis output

##### Input Parameters

- A sequence of DataElements,  $X$ , of at least 2 DataElements, in the notation for DataType  $\Psi$

##### Output Parameters

- A graph representing the periodicity in the sequence of DataElements,  $X$ , in the notation for DataType  $\mathfrak{R}$

#### TOO\_060: Chi Square Test

##### Description

Chi-square is a statistical test commonly used to compare observed data with data we would expect to obtain according to a specific hypothesis. The chi-square test is always testing the null hypothesis, which states that there is no significant difference between the expected and observed result [2].

The formula for calculating chi-square is:

$$\chi^2 = \sum_{i=1}^N \frac{(o_i - e_i)^2}{e_i} \quad (3.33)$$

That is, chi-square is the sum of the squared difference between observed ( $o$ ) and the expected ( $e$ ) data (or the deviation,  $d=o-e$ ), divided by the expected data.

#### Input Parameters

- A sequence of DataElements,  $o$ , of at least 1 DataElements in the notation for DataType  $\Psi$
- A sequence of DataElements,  $e$ , of at least 1 DataElement, in the notation for DataType  $\Psi$

#### Output Parameters

- A sequence of DataElements,  $\chi^2$ , in the notation for DataType  $\Re$

TOO\_070: Massey Berlekamp Algorithm

#### Description

Defining a connection polynomial of a general l-stage linear feedback shift-register (LFSR) as:

$$C(D) = 1 + c_1 D + c_2 D^2 + \dots c_L D^L \quad (3.34)$$

The algorithm states that for any given sequence of DataElements an L-stage polynomial can be calculated which can generate this sequence of DataElements.

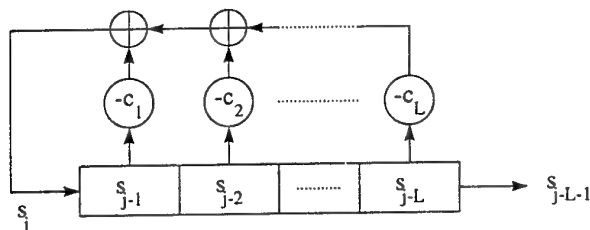


Figure 3.3: Schematic overview of linear feedback shift-register.

#### Input Parameters

- A sequence of DataElements,  $s$ , of any given length, in the notation for DataType  $\Psi$

#### Output Parameters

- A sequence of DataElements,  $c$ , of length  $L$  DataElements, in the notation for DataType  $\Psi$

TOO\_080: Polynomial Division

#### Description

A received sequence of DataElements is normally built from DataElements containing the data and from DataElements containing some kind of recursion. This recursion is used for error detection and error correction purposes. If extra DataElements are added, then this will be done according to a mathematical rule. If the received sequence of DataElements is denoted by  $C(x)$ , then this sequence is formed by a multiplication of the sequence of



DataElements and another sequence, the recursion algorithm. This second sequence is usually part of a generator matrix.

One option to get hold of both the sequence and the recursion algorithm, is to convert the sequence of DataElements into a polynomial and perform a polynomial division. i.e. find polynomial  $A(x)$  and  $B(x)$  that form  $C(x)$  as:

$$C(x) = A(x)B(x) \quad (3.35)$$

#### Input Parameters

- A sequence of DataElements,  $C$ , of any given length, in the notation for DataType  $\Psi$

#### Output Parameters

- A sequence of DataElements,  $A$ , of any given length, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $B$ , of any given length, in the notation for DataType  $\Psi$

#### 3.2.7.1.5 Pattern Statistics ...

Table 3.10: Pattern Statistics

Pattern Statistics	Shortcut
Real Periodic Pattern Search	
Semi Periodic Pattern Search	
Aperiodic Pattern Search	
Pattern Distance Statistics	
Symbol Statistics	

Many kinds of patterns exist in communication DataElements sequences. Many of these are used for synchronisation, but the existence of certain patterns is also a sign for specific information repeated over and over in the sequence of DataElements. The following functions form a tool, for an analyst to overview large quantities of data.

For this function, it is necessary to find a histogram of the number of hits compared to the code sequence and the found distances between the various search hits. Therefore a 3D-graphic display shall be created, as shown in Figure 3.4. Subsequently the analyst must be able to perform automated or manual analysis to obtain the different searches:

- Real Periodic Pattern Search,
- Semi Periodic Pattern Search,
- Aperiodic Pattern Search.

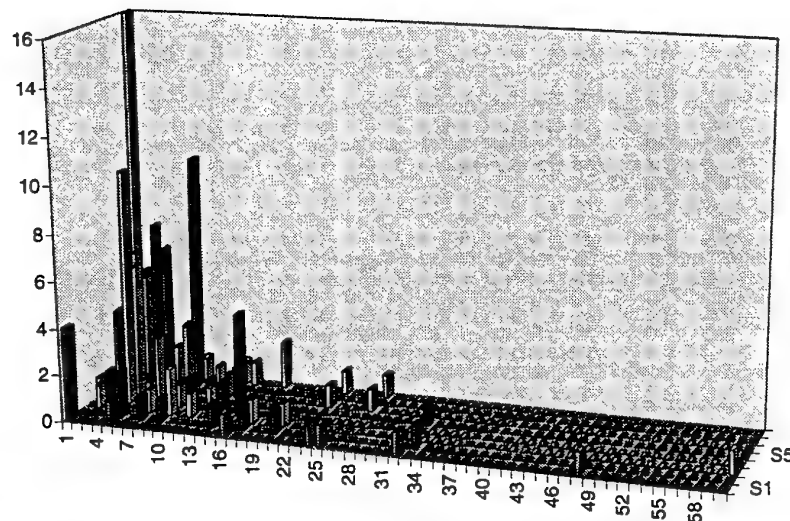


Figure 3.4: Example of pattern statistics analysis.

After examination of various code lengths a correlation should be performed to find the similarities between the different histograms.

The following statistical functions on patterns shall be provided for this:

#### TOO\_090: Real Periodic Pattern Search

##### Description

After the global search the resulting information is to be examined for a number of hits  $n$  for various patterns. If a certain code occurs many times in the selected code sequence with a deviation  $\delta$  in the found distances, then a real periodic code sequence has been found.

##### Input Parameters

- A sequence of DataElements,  $A$ , of any given length, in the notation for DataType  $\Psi$
- $\delta$ : deviation in distances from keyboard
- $n$ : from keyboard

##### Output Parameters

- A graph, which might be correlated with other graphs to be interpreted by the analyst.

#### TOO\_100: Semi Periodic Pattern Search

##### Description

A Semi Periodic Pattern occurs on distances which are (nominally) a multiple of a common number of DataElements.

##### Input Parameters

- A sequence of DataElements,  $A$ , of any given length, in the notation for DataType  $\Psi$

##### Output Parameters

- A graph, which might be correlated with other graphs to be interpreted by the analyst

#### TOO\_110: Aperiodic Pattern Search

##### Description

After the overall search the resulting information is to be examined for relatively large numbers of hits for various patterns. If a certain code occurs many times in the selected code sequence but with varying distances, then an isolated aperiodic code sequence has been found.

#### Input Parameters

- A sequence of DataElements,  $A$ , of any given length, in the notation for DataType  $\Psi$

#### Output Parameters

- A graph, which might be correlated with other graphs for a "Common Distances Diagram", but which needs to be interpreted by the analyst

#### TOO\_115: General Pattern Search

- It shall be possible to search for any user defined pattern
- All occurrences of a found pattern shall be highlighted
- It shall be possible to define at least 8 different search patterns
- Different patterns shall be highlighted in different, distinguishable ways
- The maximum number of DataElements for a search pattern shall be at least 1024

#### TOO\_120: Pattern Distance Statistics

##### Description

From a certain selected pattern,  $P$ , in a sequence of DataElements,  $S$ , all the locations,  $l$ , and the interval distances,  $d$ , between the next occurring pattern will be filed and various statistics, average and standard deviation, will be calculated.

##### Input Parameters

- A sequence of DataElements,  $S$ , of any given length in the notation for DataType  $\Psi$
- A pattern,  $P$ , of any given length, in the notation for DataType  $\Psi$

##### Output Parameters

- A table,  $T$ , of a given length, from presentation  $\aleph$ , containing the pattern locations
- The average interval distance,  $\bar{d}$ , rounded to presentation  $\aleph$
- The standard deviation,  $\sigma$ , rounded to presentation  $\aleph$

#### TOO\_130: Symbol Statistics

##### Description

In order to prevent a receiver from misunderstanding the received DataElements, many transmission systems add information to the transmitted sequence of DataSymbols. Transmission systems often use a system called 'bitstuffing'. The tool inserts a sequence of DataElements in case an other predefined sequence appears in the sequence to be transmitted. This procedure can be performed with any kind of sequence and is not limited to single DataElements.

##### Input Parameters

- A sequence of DataElements,  $T$ , of any given length, in the notation for DataType  $\Psi$ , to be transmitted
- A sequence of DataElements,  $S$ , of any given length, in the notation for DataType  $\Psi$ , to be stuffed

##### Output Parameters

- location of the various patterns

- length of the various patterns
- frequency of the various patterns

#### 3.2.7.1.6 Coding scheme analysis

There are many different types of codes and the optimal code for a given set of channel conditions may not resemble the optimal code for another set of channel conditions. It is probably true that starting with any given code, one could devise a set of channel conditions for which that code is optimal. Many devised channel conditions are extremely unlikely in practice. However, if attention is restricted to the channels that are most often encountered, there are five overlapping code classes which are encountered most often:

1. Convolutional codes,
2. Short binary block codes,
3. Reed-Solomon codes,
4. Concatenated codes,
5. Codesigned coding and modulation schemes.

These functions will be able to generate the coding scheme given the DataElements.

TOO\_140: Convolutional Code Analysis

##### Description

Convolution codes are characterised by their suitability for analogue engineering and their high performance, especially when they can be coded by the Viterbi algorithm (VA). The key to their success is the VA's ability to process "soft-decision" data or reliability information into maximum-likelihood estimates of the transmitted sequence. If these reliabilities are output in addition to data, the combined output is called soft-decision data. Output data is built by means of a Forward Shift Register with multiple output options, as given by Figure 3.5. Convolution Codes are often combined with interleaving as a form of randomising burst errors to optimise the code sequence to Convolution Codes.

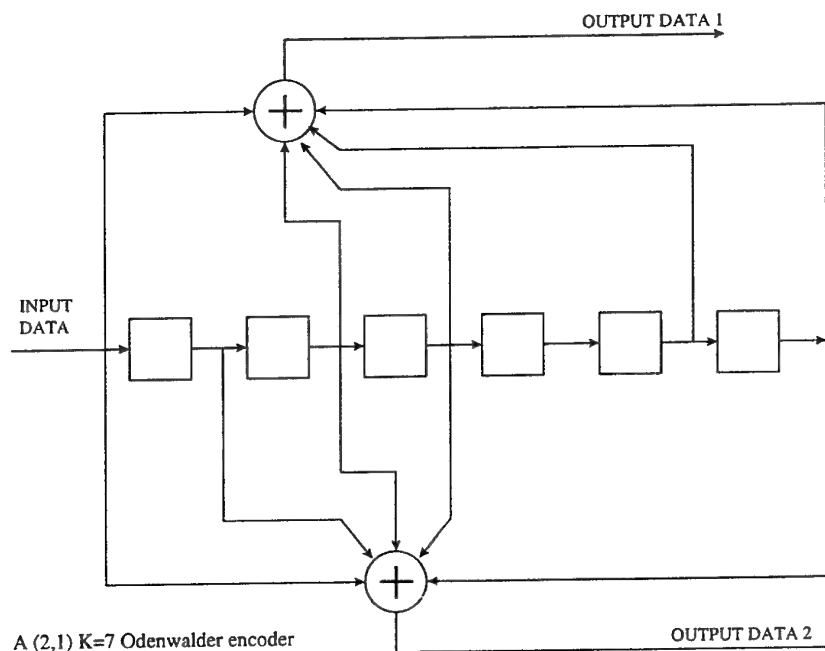


Figure 3.5: Example of a Convolution Encoder

**Input Parameters**

- A sequence of input DataElements, of any given length, in the notation for DataType  $\Psi$ .

**Output Parameters**

- The polynomial presentations of the interconnection between the registers and the soft-decision data sequences, in the notation for DataType  $\Psi$ .

**TOO\_150: Short Block Code Analysis****Description**

A *block code* over  $A$  is a subset of  $A^N$ , where  $A$  is a pre-set alphabet and  $N$  is an integer referred to as the length of the code. If  $C$  denotes the subset of  $A^N$  ( $C \subseteq A^N$ ), the elements of  $C$  are referred to as codewords. If  $A$  has the structure of a field and  $C$  is a vector space of dimension  $K$  over  $A$ ,  $C$  is called a linear  $(N,K)$  code over  $A$ . If  $A$  is the binary field  $GF(2)$ , the code is called a binary code. If  $A$  is a field with more than two elements, the code is called a symbol code. If a basis for  $GF(2^m)$  over  $GF(2)$  is chosen, linear binary codes are obtained from linear symbol codes over  $GF(2^m)$ .

**Input Parameters**

- A sequence of codes,  $C$ , in the notation for DataType  $\Psi$

**Output Parameters**

- A value,  $N$ , corresponding to the size of the field  $A$ , in the notation for DataType  $\aleph$
- A value,  $K$ , corresponding to the length of the code, in the notation for DataType  $\aleph$
- The Generator Matrix

**TOO\_160: Reed-Solomon Code Analysis****Description**

Reed-Solomon (RS) codes are a class of symbol codes for which encoding and bounded-distance algorithms are practical to implement. The parameters of RS codes follow. Let  $b$  be the symbol size in  $m$  DataElements. A  $RS(N, K)$  symbol code over  $GF(m^b)$  exists whenever:

$$0 < K < N \leq m^b + 1 \quad (3.36)$$

The minimum Hamming distance of a  $RS(N, K)$  code is

$$d_{\min} = N - K + 1 \quad (3.37)$$

The code is generated by means of a generator matrix. As a result this function generates the parameters that were responsible for the creation of the sequence of DataElements.

#### Input Parameters

- A sequence of DataElements, in the notation for DataType  $\Psi$

#### Output Parameters

- The parameters  $m, b, N, K$  and a generator matrix

TOO\_170: Concatenated Code Analysis

#### Description

Concatenated codes use two or more decoders in series. The *inner code* is the code nearest to the channel; hence the signal is first processed by the inner decoder. The outer code is the last decoder to process the signal. This is illustrated in Figure 3.6. This function shall be able to detect which coders are concatenated together.

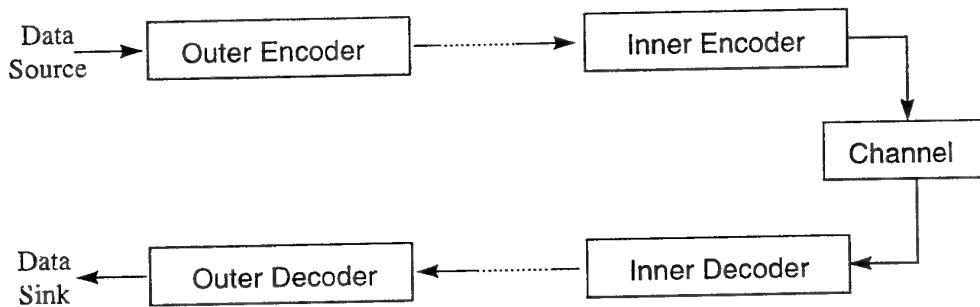


Figure 3.6: Schematic view of a concatenated Coder

#### Input Parameters

- A sequence of DataElements, in the notation for DataType  $\Psi$

#### Output Parameters

- A list of coders

TOO\_180: Codesigned Coding and Modulation Scheme Analysis

#### Description

Previous Coding Schemes are 'active' on DataElements independent of the physical channel characteristics only. Modern communication systems are a mingled form of DataElement coding and modulation scheme 'coding'. The rapid acceptance of trellis-coded modulation (TCM) has been one of the more spectacular success stories of coding technology. Most TCM schemes operate on the same principles, following five major steps:

1. Selecting a block lattice.

2. Selecting a finite number of points from the lattice as a set of modulation points; this is sometimes called the *constellation*.
3. Labelling selected points to insure that the Euclidean distance between distinct points is related to the difference in the labels.
4. Selecting a convolutional encoder that generates a succession of point labels; in other words, a sequence of constellation points is transmitted that includes code structure.
5. Decoding and demodulation via the Viterbi Algorithm (VA).

**Input Parameters**

- A sequence of DataElements, from presentation A, which includes modulation information

**Output Parameters**

- A sequence of DataElements, from an n-ary presentation  $\Psi$
- The type of TCM coding
- Number of trellis points, frequency and phase locations

TOO\_190:      Hamming Weight Analysis

**Description**

This function's result is the number of non-zero elements in the selected Sequence of Code Symbols.

$$w(x) = |\{ i : x_i \neq \text{NullSymbol} : 1 \leq i \leq n \}| \quad (3.38)$$

**Input Parameters**

- A sequence of DataElements,  $x$ , of any given length, in the notation for DataType  $\Psi$

**Output Parameters**

- A number representing the number of non-zero DataElements in  $x$

TOO\_200:      Hamming Distance Analysis

**Description**

Hamming distance analysis is used to check the numerical 'distance' for a certain coding used in the received sequence of DataElements. Let

$$d(x, y) = |\{ i : x_i \neq y_i : 1 \leq i \leq n \}| \quad (3.39)$$

That is,  $d(x, y)$  is the number of positions in which  $x$  and  $y$  differ.

**Input Parameters**

- A sequence of DataElements,  $x$ , of any given length, in the notation for DataType  $\Psi$
- A sequence of DataElements,  $y$ , of the same length as  $x$ , in the notation for DataType  $\Psi$

**Output Parameters**

- A number representing the number of different positions between  $x$  and  $y$

TOO\_210:      CRC Analysis

**Description**

Cyclic Redundancy Check is a way of telling whether or not a sequence of DataElements has suffered errors. The generated 'extra' DataElements are inserted into the original sequence. Afterwards the received sequence is divided by some generator polynomial. If the result of this division is zero then 'no error has occurred', otherwise one or more positions contain an error.

**Input Parameters**

- A sequence of DataElements,  $x$ , increased with the number of redundant DataElements, in the notation for DataType  $\Psi$

**Output Parameters**

- The type of CRC code, i.e. the generator polynomial, from an  $n$ -ary presentation  $\Psi$
- A sequence of DataElements,  $x$ , of any given length, in the notation for DataType  $\Psi$

TOO\_220: BCH Analysis

**Description**

BCH-Coding or 'Bose-Chaudhuri-Hocquenghem-Coding' is a block code. A cyclic code of length  $n$  over  $GF(q)$  is a *BCH code of designed distance  $\delta$*  if, for some integer  $b \geq 0$ ,

$$g(x) = l.c.m. \{ M^{(b)}(x), M^{(b+1)}(x), \dots, M^{(b+\delta-2)}(x) \} \quad (3.40)$$

I.e.  $g(x)$  is the lowest degree monic polynomial over  $GF(q)$  having  $\alpha^b, \alpha^{b+1}, \dots, \alpha^{b+\delta-2}$  as zeros. Therefore  $c$  is in the code if

$$c(\alpha^b) = c(\alpha^{b+1}) = \dots = c(\alpha^{b+\delta-2}) = 0 \quad (3.41)$$

Hence the code has a string of  $\delta-1$  consecutive powers of  $\alpha$  as zeros. The minimum distance is greater than or equal to the designed distance  $\delta$ . A parity check matrix  $H$  is designed for the code, according to

$$H = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \dots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \dots & \alpha^{(n-1)(b+1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \alpha^{b+\delta-2} & \dots & \dots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix} \quad (3.42)$$

**Input Parameters**

- A sequence of DataElements,  $X$ , of any given length, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of DataElements,  $Y$ , in the notation for DataType  $\Psi$
- A polynomial  $g(x)$ , in the notation for DataType  $\Psi$
- A value  $b$ , in the notation for DataType  $\aleph$

TOO\_230: Viterbi Analysis

**Description**

This is one end of the extremities of convolution coding. It is called a *maximum-likelihood code*. It achieves the optimal performance, but requires extensive hardware for computation and hardware. This in contrast to *feedback coding*, which sacrifices performance in exchange for simplified hardware. A maximum-likelihood coder needs to examine an entire sequence of DataElements and find a valid path that has the smallest Hamming distance from this sequence, out of a total of  $2^N$  paths as shown in Figure 3.7. A Viterbi decoder must calculate two metrics for each node in the path, and store  $2^{kL}$  surviving paths (minimal Hamming distance), each consisting of  $N$  branches. Hence  $L$ , the code length and  $k$  have to be declared.



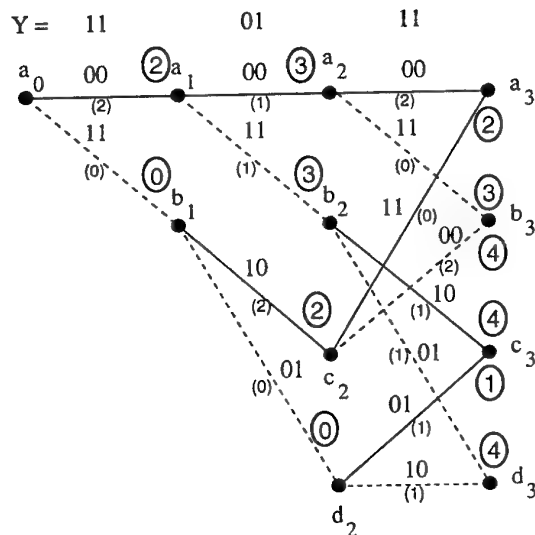


Figure 3.7: Example of a received Viterbi Code

**Input Parameters**

- A sequence of DataElements,  $Y$ , of any given length, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of DataElements,  $X$ , in the notation for DataType  $\Psi$
- A length of the Viterbi coder,  $L$
- A number to define the capacity of the Viterbi coder of the surviving paths

## TOO\_240: Stuffing Analysis

**Description**

Extra CodeSymbol(s) need to be inserted into a sequence of DataElements to prevent certain predefined sequences to arise. Actually the number of available DataSequences is reduced by means of such a coding schema.

**Input Parameters**

- A sequence,  $s$ , the 'stuffed' sequence of DataElements, in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence,  $c$ , which describes the sequence of DataElements which should not occur, in the notation for DataType  $\Psi$
- A sequence,  $s$ , the initial sequence of DataElements, in the notation for DataType  $\Psi$

## TOO\_250: Tree Code Analysis

**Description**

This function shall be able to detect the existence of 'Tree Codes', like Huffman Coding. 'Tree Codes' have the common property that the probabilities for certain DataElement is related to the code length of the representing sequence of DataElements.

**Input Parameters**

- A sequence of DataElements,  $x$ , of any given length, in the notation for DataType  $\Psi$

**Output Parameters**

- A parameter value indicating whether a variable codelength coding has been used

- if applicable, the Codes representing the coding scheme

### 3.2.7.2 Maintain Verification Functions ...

TOO\_255: It may be necessary to maintain the *verification functions*. These functions need to be programmed in a standard programming language in which special libraries are allowed. Maintenance shall at least include:

- Create
- Edit
- Delete

### 3.2.7.3 Synthesis Functions ...

Table 3.11: Synthesis Functions

Synthesis Functions	Shortcut
Feedback Shift Register Synthesis	
Code Scheme Synthesis...	
CRC Synthesis	
BCH Synthesis	
Viterbi Synthesis	
Stuffing	
Edit Coding Table ...	

#### 3.2.7.3.1 Feedback Shift Register

TOO\_260:

##### Description

Feedback Shift Registers are generally used to produce a pseudo random sequence of DataElements, as shown in Figure 3.8, which are to be processed along with DataElements to come to a sequence of DataElements which has a minimum of periodicity. Feedback Shift Registers are described by three parameters:

- length of the Feedback Shift Register
- the sequence which describes the 'feedback wiring'
- the sequence with the initial filling of the register
- A possible fourth parameter is the number of DataElements the Feedback Shift Register needs to generate.

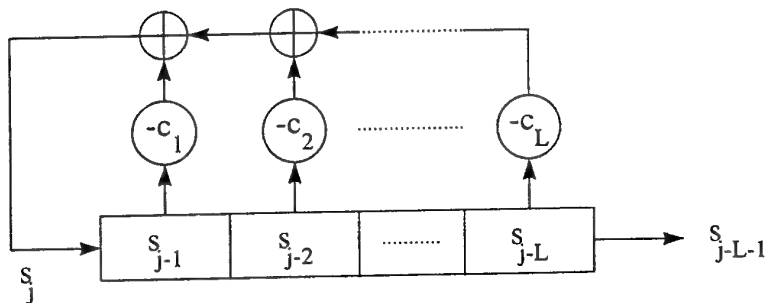


Figure 3.8: Model of a linear feedback shift-register.

#### Input Parameters

- The length,  $L$ , of the Feedback Shift Register, from presentation  $\aleph$
- A sequence,  $c$ , which describes the 'feedback wiring', in the notation for DataType  $\Psi$
- A sequence,  $s$ , with the initial filling of the register, in the notation for DataType  $\Psi$

#### Output Parameters

- A sequence,  $s$ , in the notation for DataType  $\Psi$

#### 3.2.7.3.2 Code Scheme Building

There are many different types of codes and the optimal code for a given set of channel conditions may not resemble the optimal code for another set of channel conditions. It is probably true that starting with any given code, one could devise a set of channel conditions for which that code is optimal. From these codes, many devised channel conditions are extremely unlikely in practice. We restrict ourselves to the channels that are most often encountered, following the most probable five overlapping code classes most commonly used:

1. Convolutional codes,
2. Short binary block codes,
3. Reed-Solomon codes,
4. Concatenated codes,
5. Codesigned coding and modulation schemes.

This paragraph handles function descriptions that are able to deal with these codes, i.e. are able to generate the DataElements given the coding scheme. The functions results will be available for other functions for further analysis. It's noted that the majority of coding schemes are subgroups of one of the above given five codeclasses.

#### TOO\_280: Convolutional Code Building

##### Description

Convolution codes can be characterised by their high performance whenever they can be decoded by the Viterbi algorithm (VA). The key to their success is the VA's ability to process "soft-decision" data or reliability information into maximum-likelihood estimates of the transmitted sequence. In case these reliabilities are output in addition to data, the combined output is called soft-decision data. Output data is built by means of a Forward Shift Register with multiple output options, as shown in Figure 3.9. Convolution Codes are often combined with interleaving to obtain uniform distributed errors to optimise the code sequence to Convolution Codes.

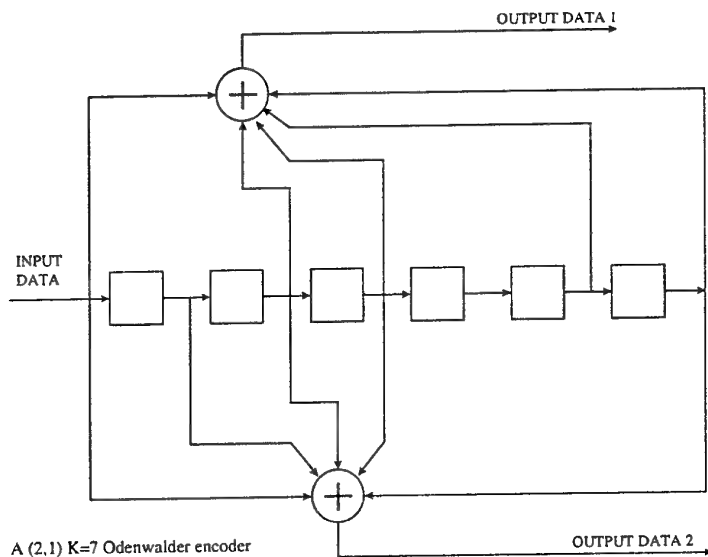


Figure 3.9: Example of a convolution encoder.

**Input Parameters**

- A sequence of input DataElements, of any given length, in the notation for DataType  $\Psi$
- The polynomial presentations of the interconnection between the registers in the notation for DataType  $\Psi$

**Output Parameters**

- A sequence of output DataElements, in the notation for DataType  $\Psi$

**TOO\_290: Short Block Code Building****Description**

A *block code* over  $A$  is a subset of  $A^N$ , where  $A$  is a pre-set alphabet and  $N$  is an integer referred to as the length of the code. If  $C$  denotes the subset of  $A^N$  ( $C \subseteq A^N$ ), the elements of  $C$  are referred to as codewords. If  $A$  has the structure of a field and  $C$  is a vector space of dimension  $K$  over  $A$ ,  $C$  is called a linear  $(N,K)$  code over  $A$ . If  $A$  is the binary field  $GF(2)$ , the code is called a binary code. If  $A$  is a field with more than two elements, the code is called a symbol code. If a basis for  $GF(2^m)$  over  $GF(2)$  is chosen, linear binary codes are obtained from linear symbol codes over  $GF(2^m)$ .

**Input Parameters**

- A value corresponding to the size of the field  $A$ , in the notation for DataType  $\aleph$
- A value corresponding to the length of the code, in the notation for DataType  $\aleph$
- A sequence of messages

**Output Parameters**

- A sequence of codes, in the notation for DataType  $\Psi$

**TOO\_300: Reed-Solomon Code Building****Description**

Reed-Solomon (RS) codes are a class of symbol codes for which encoding and bounded-distance algorithms are practical to implement. Let  $b$  be the symbol size in  $m$  DataElements. An  $RS(N,K)$  symbol code over  $GF(m^b)$  exists if:

$$0 < K < N \leq m^b + 1 \quad (3.43)$$

The minimum Hamming distance of an  $RS(N,K)$  code is given by

$$d_{\min} = N - K + 1 \quad (3.44)$$

The code is generated by means of a generator matrix.

#### Input Parameters

- The parameters  $m, b, N, K$  and a generator matrix
- A sequence of messages

#### Output Parameters

- A sequence of DataElements, in the notation for DataType  $\Psi$

### TOO\_310: Concatenated Code Building

#### Description

Concatenated codes use two or more decoders in series. The *inner code* is the code nearest to the channel; i.e. the signal is first processed by the inner decoder. The outer code is the last decoder to process the signal. See figure. This function is able to concatenate various coders together.

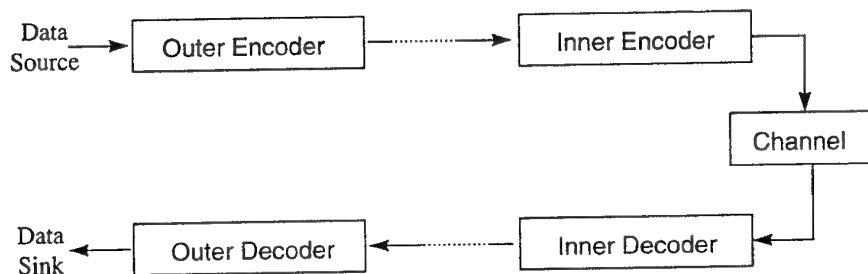


Figure 3.10: Concatenated Code Schema.

#### Input Parameters

- A list of coders, with their own set of parameters

#### Output Parameters

- A sequence of DataElements, in the notation for DataType  $\Psi$

### TOO\_320: Codesigned Coding and Modulation Scheme Building

#### Description

Previous Coding Schemes are 'active' on DataElements only, independent of the physical channel characteristics. Modern communication systems are a mingled form of DataElement coding and modulation scheme 'coding'. The rapid acceptance of trellis-coded modulation (TCM) has been one of the more spectacular achievements of coding technology. Most TCM schemes operate on the same principles, following five major steps:

1. Selecting a block lattice.
2. Selecting a finite number of points from the lattice as a set of modulation points; this is also called the *constellation*.
3. Labelling selected points to insure that the Euclidean distance between distinct points is related to the difference in the labels.

4. Selecting a convolutional encoder that generates subsequential point labels; in other words, a sequence of constellation points is transmitted that includes code structure.
5. Decoding and demodulation via the Viterbi Algorithm (VA).

#### Input Parameters

- A sequence of DataElements, in the notation for DataType  $\Psi$
- The type of TCM coding
- Number of trellis points, frequency and phase locations

#### Output Parameters

- A sequence of DataElements, in the notation for DataType  $\Im$ , which includes modulation information

#### 3.2.7.3.3 CRC Synthesis

TOO\_330:

##### Description

Cyclic Redundancy Check is a way of identifying whether or not a sequence of DataElements has suffered erroneous Positions. The generated 'extra' DataElements are inserted into the original sequence. Afterwards the received sequence is divided by some generator polynomial. In case the result of this division is zero then 'no error has occurred', otherwise one or more positions contain an error.

##### Input Parameters

- The type of CRC code, i.e. the generator polynomial, from an n-ary presentation  $\Psi$
- A sequence of DataElements,  $x$ , of any given length, in the notation for DataType  $\Psi$

##### Output Parameters

- A sequence of DataElements,  $x$ , increased with the number of redundant DataElements, in the notation for DataType  $\Psi$

#### 3.2.7.3.4 BCH Synthesis

TOO\_340:

##### Description

BCH-Coding or 'Bose-Chaudhuri-Hocquenghem-Coding' is a block code. A cyclic code of length  $n$  over  $GF(q)$  is a *BCH code of designed distance  $\delta$*  if, for some integer  $b \geq 0$ ,

$$g(x) = l.c.m. \{ M^{(b)}(x), M^{(b+1)}(x), \dots, M^{(b+\delta-2)}(x) \} \quad (3.45)$$

(l.c.m. = least common multiplier)

I.e.  $g(x)$  denotes the lowest degree monic polynomial over  $GF(q)$  having  $\alpha^b, \alpha^{b+1}, \dots, \alpha^{b+\delta-2}$  as zeros. Therefore,  $c$  is in the code if

$$c(\alpha^b) = c(\alpha^{b+1}) = \dots = c(\alpha^{b+\delta-2}) = 0 \quad (3.46)$$

Hence the code has a string of  $\delta-1$  consecutive powers of  $\alpha$  as zeros. The minimum distance is greater than or equal to the designed distance  $\delta$ . A parity check matrix can be designed for this code as follows

$$H = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \dots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \dots & \alpha^{(n-1)(b+1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \alpha^{b+\delta-2} & \dots & \dots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix}. \quad (3.47)$$

#### Input Parameters

- A sequence of DataElements,  $Y$ , of any given length, from an  $n$ -ary presentation  $\Psi$
- A polynomial  $g(x)$ , in the notation for DataType  $\Psi$
- A value  $b$ , from presentation  $\aleph$

#### Output Parameters

- A sequence of DataElements,  $X$ , of any given length, in the notation for DataType  $\Psi$

#### 3.2.7.3.5 Viterbi Synthesis

TOO\_350:

##### Description

Actually this is one end of the extremities of the set of convolution codes, that has best results. It's called a *maximum-likelihood code*. It achieves the optimal performance, but requires extensive hardware for computation and hardware. This in contrast to *feedback coding*, which sacrifices performance in exchange for simplified hardware. A maximum-likelihood coder needs to examine an entire sequence of DataElements and find a valid path that has the smallest Hamming distance from this sequence, out of a total of  $2^N$  paths. This would be computationally unacceptable. A Viterbi decoder must calculate two metrics for each node in the path, and store  $2^{kl}$  surviving paths (minimal Hamming distance), each consisting of  $N$  branches. Hence  $L$ , the code length and  $k$  have to be declared.

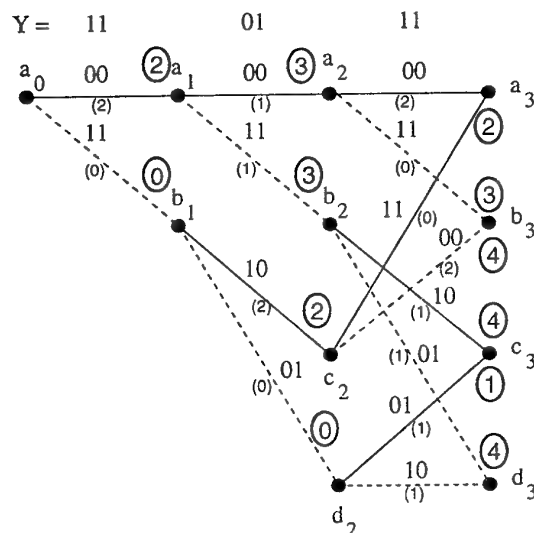


Figure 3.11: Example of a received Viterbi Code

#### Input Parameters

- A sequence of DataElements,  $X$ , of any given length, in the notation for DataType  $\Psi$
- A length of the Viterbi coder,  $L$

- A number to define the capacity of the Viterbi coder to define the surviving paths

#### **Output Parameters**

- A sequence of DataElements,  $Y$ , of any given length, in the notation for DataType  $\Psi$

#### **3.2.7.3.6 Stuffing Synthesis**

TOO\_360:

##### **Description**

One or more CodeSymbols need to be inserted into a sequence of DataElements to prevent certain predefined sequences to arise. Actually the number of available DataSequences is reduced by means of such a coding schema.

##### **Input Parameters**

- A sequence,  $c$ , which describes the sequence of DataElements not to occur, in the notation for DataType  $\Psi$
- The Stuffing Pattern in the notation for DataType  $\Psi$
- A sequence,  $s$ , the initial sequence of DataElements, in the notation for DataType  $\Psi$

##### **Output Parameters**

- A sequence,  $s$ , the 'stuffed' sequence of DataElements, in the notation for DataType  $\Psi$

#### **3.2.7.3.7 Code Table Synthesis**

TOO\_370: It shall be possible to code/decode any selection of DataElements using one of the following coding tables:

- ASCII
- ITU-2
- ITU-5
- Non Latin alphabets
- Cyrillic
- UNICODE
- Binary
- Octal
- Decimal
- Hexadecimal
- n-positions
- user defined coding table

TOO\_371: It shall be possible to interpret DataElements as sound samples, and create a .WAV file of these samples.

TOO\_372: It shall be possible to interpret DataElements as video samples, and create a graphics file of these samples

#### **3.2.7.3.8 Edit Coding Table**

TOO\_380: There shall be a means to create and edit user defined coding tables.

TOO\_390: The size of the coding table shall be selectable by the operator.

TOO\_400: Entries for the coding table shall be generated by the system.

TOO\_410: Translation codes shall be editable by the operator.

TOO\_420: It shall be possible to select a default for the translation codes.



TOO\_430: The use of multiple shift codes shall be supported.

#### 3.2.7.4 Maintain Synthesis Functions

TOO\_440: It may be necessary to maintain the *synthesis functions*. These functions need to be programmed in a standard programming language in which special libraries are allowed. Maintenance shall at least include:

- Create
- Edit
- Delete

#### 3.2.7.5 Macros

Table 3.12: Macros

Macros	Shortcut
Macro1	
Macro2	

TOO\_449: There shall be a facility for the selection of Macros.

#### 3.2.7.6 Maintain Macros

TOO\_450: It may be necessary to maintain the Macros. Maintenance shall at least include:

- Create
- Edit
- Delete

#### 3.2.8 Window Functions (WIN\_)

Table 3.13: Window menu

Window	(Shortcut)
New Window	
Cascade	
Tile	
Arrange Icons	
1 Display-a	
√ 2 Display-b	

WIN\_010: It shall be possible to create new windows.

WIN\_020: It shall be possible to show the windows cascaded.

WIN\_030: It shall be possible to show the windows tiled.

WIN\_040: It shall be possible to arrange iconised windows.

### 3.2.9 Help (HEL\_)

Table 3.14: Help menu

Help	(Shortcut)
BIAS Help Topics	
About BIAS	

- HEL\_010: There shall be a context sensitive help function.  
HEL\_020: There shall be a list of help topics selectable by the operator.  
HEL\_030: It shall be possible for the user (system manager) to add new help topics.

## 3.3 CSCI external interface requirements (EIR\_)

Connections have to be made between a possible modulator/demodulator setup and the BIAS system. The present modulator systems are capable of presenting a n-ary signal with an analogue signal level. These levels can be converted to a number of bits or bit streams. Future systems have to present the data in a kind of a data structure to come to a representation that includes the actual data, information about the kind of data that is received and an analogue representation of the received data. All this information together is necessary to come to an unambiguous view of the data. Other forms of interfacing will be more or less straight forward.

### 3.3.1 Interface identification and diagrams

BIAS has external interfaces to:

1. Receiver output (via Demodulator)
2. Existing data handling systems
3. External resources, e.g. from Intelligence Services
4. Storage media
5. Printers

### 3.3.2 Interface with Receiver/Demodulator

- EIR\_010: BIAS shall accept input from a receiver/demodulator for DataTypes as defined in section 3.5.1.3. The channels shall have a form of synchronisation to make sure that the various events can be lined up in time.  
EIR\_011: There shall be at least 2 receiver/demodulator inputs available (for a forward channel and a return channel ).  
EIR\_020: The maximum number of lines per inputs shall be at least 16.  
EIR\_030: The maximum number of levels per line shall be at least 4.

### 3.3.3 Interface with existing Data Handling System

- EIR\_050: The client is working with a data distribution and storage system which distributes analogue data through an analogue switch board to different demodulators and various interception units. These units are trained for various tasks. One of these tasks is the analysis of newly received unknown communications systems. An interface shall exist for this purpose.

### 3.3.4 Interface with External Resources and Storage Media

EIR\_060: It shall be possible to store extra information available from the demodulator in the same way as it is received from the demodulator. Also it shall be possible to store the analogue representation of the received signal.

### 3.3.5 Interface with Printers

EIR\_070: Output to printers shall be performed using standard printer drivers.

## 3.4 CSCI internal interface requirements

The description of the various internal interface requirements is dependent on the kind of implementation that is going to be used for BIAS.

## 3.5 CSCI internal data requirements

The description of the various internal data requirements is dependent on the kind of implementation that is going to be used for BIAS.

### 3.5.1 Requirements (DAT\_)

The following definitions are applicable. They shall be included in BIAS.

#### 3.5.1.1 InputData

DAT\_005: InputData describes the input from a receiver (via a demodulator) as the level of the input signal plus a time mark indicating the moment of change to this level.

#### 3.5.1.2 DataElements

DAT\_010: DataElements describe the input signal for a normalised time period. This time period shall be derived from the calculated Baudrate and being the Signal Element Timing. Non standard length elements (e.g. asynchronous stop bits) shall be normalised to the standard length.

#### 3.5.1.3 DataTypes

DAT\_020: DataElements can be any of the following DataTypes:

1. single line/channel
  - 2-level: BINARY
  - n-level: nARY
2. multiple (m-)line/channel
  - 2-level: mBINARY
  - n-level: mnARY

#### 3.5.1.4 States of DataTypes

DAT\_025: Apart from the states representing DataTypes the following extra miscellaneous states shall be defined:

- unknown status
- random status
- empty status
- noise status

DAT\_040: Each of these representations shall further allow for the indication of a masked status and non standard length.

#### 3.5.1.5 DataCodes

DAT\_030: DataCodes shall allow for the representation of all states of all DataTypes.

DAT\_050: All operations shall be performed on DataElements or DataCodes.

#### 3.5.1.6 DataSymbols

DAT\_060: For the purpose of on screen and hard copy visualisation each DataCode shall be mapped on a DataSymbol.

DAT\_070: A DataSymbol can be:

- a flox
- a fixed width symbol

both in a selectable font type and font size.

DAT\_080: It shall be possible to “zoom in” into a multi level DataSymbol to obtain a detailed view of the contents of each DataSymbol. In that case each DataSymbol shall be split into as much DataSymbols as the the number of separate input lines.

#### 3.5.2 Constraints

NA

### 3.6 Adaptation requirements

#### 3.6.1 Installation-dependent data

NA

#### 3.6.2 Operational parameters

NA

### 3.7 Safety requirements

NA

### **3.8 Security and privacy requirements**

NA

### **3.9 CSCI environment requirements**

The description of the CSCI environment requirements is dependent on the kind of implementation that is going to be used for BIAS.

### **3.10 Computer resource requirements**

The description of the Computer resource requirements is dependent on the kind of implementation that is going to be used for BIAS.

#### **3.10.1 Computer hardware requirements**

NA

#### **3.10.2 Computer hardware resource utilisation requirements**

NA

#### **3.10.3 Computer software requirements**

NA

#### **3.10.4 Computer communications requirements**

NA

### **3.11 Software quality factors**

The description of the Software quality factors is dependent on the kind of implementation that is going to be used for BIAS.

### **3.12 Design and implementation constraints**

The description of the Design and implementation constraints is dependent on the kind of implementation that is going to be used for BIAS.

### **3.13 Personnel-related requirements (HER\_)**

The following main human engineering requirements are applicable:

HER\_010: the system shall be windows oriented

HER\_020: all operations shall be performed recursively

- HER\_030: there may be no limitation to the number of simultaneous presentations
- HER\_040: standard functions shall be performed in a way as usual in a windows environment
- HER\_050: there shall be an on line context sensitive help function
- HER\_060: the user interface shall be entirely in the English language
- HER\_070: There shall be available a set of 'work charts', describing standard procedures to be performed by operators.

### **3.14 Training-related requirements**

NA

### **3.15 Logistics-related requirements**

NA

### **3.16 Other requirements**

NA

### **3.17 Packaging requirements**

NA

### **3.18 Precedence and criticality of requirements**

NA

## **4. Qualification provisions**

### **4.1 Qualification methods**

NA

### **4.2 Special qualification requirements**

NA

## 5. Requirements traceability

NA



## 6. Notes

### 6.1 Abbreviations

BIAS	BIt Analyse System
CSCI	Computer Software Configuration Item
MIL-STD-498	Military Standard 498

### 6.2 Definitions

InputData	see 3.5.1.1
DataElement	see 3.5.1.2
DataType	see 3.5.1.3
DataCode	see 3.5.1.5
DataSymbol	see 3.5.1.6
Description Language	abstract recursive language, describing a communications system
flox	a graphical presentation of CodeSymbols
$GF(\mathbb{K}^+)$	representation of a Galois Field in $\mathbb{K}^+$
$\{\mathbb{R}\}$	value from the set of reals

### 6.3 Items to be defined

NA

### 6.4 General information

NA

## Appendix A Description Language

A possible definition for the Description Language is given below.  
Enhancements to this language are welcomed.

### Syntax

	=	'or'
,	=	delimiter
{ ... }	=	optional item
[n m ... z]	=	exactly n, or m,...,or z, with n, m, z > 0
[n..m]	=	minimum n and maximum m, with m > n
[n+]	=	n or more
< ... >	=	exact specification, list of parameters
@	=	reference
Field	=	any value of the Galois Field $GF(K^+)$
'a'	=	literal, exactly specified Field
Structure	=	Field{, Structure}   'a' {, Structure}   Structure {, Structure}

### Keywords

**A = COPY b <n>**

The discrete structure A is built from n identical discrete substructures b (n > 2)

**A = INVERT b[n]**

**A = INVERT b[n] <b<sub>x</sub>,b<sub>y</sub>,...,b<sub>z</sub>>**

**A = INVERT b[n] <@List>**

The discrete structure A is built from the n discrete substructures b<sub>1</sub>..b<sub>n</sub> of the discrete structure b[n] by inverting all discrete substructures of b[n], or only the discrete substructures of b[n] specified in <>

**A = AND b[n]**

The structure **A** is built from the **n** substructures  $b_1..b_n$  of the structure **b[n]** by performing a discrete AND operation.

**A = OR b[n]**

The discrete structure **A** is built from the **n** discrete substructures  $b_1..b_n$  of the discrete structure **b[n]** by performing a discrete OR operation.

**A = XOR b<sub>1</sub>, b<sub>2</sub>**

The discrete structure **A** is built from the discrete substructures **b<sub>1</sub>** and **b<sub>2</sub>** of the discrete structure **b[n]** by performing a discrete Exclusive OR operation.

**A = NAND b[n]**

The discrete structure **A** is built from the **n** discrete substructures  $b_1..b_n$  of the discrete structure **b[n]** by performing a discrete Inverted AND operation.

**A = NOR b[n]**

The discrete structure **A** is built from the **n** discrete substructures  $b_1..b_n$  of the discrete structure **b[n]** by performing a discrete Inverted OR operation.

**A = XNOR b<sub>1</sub>, b<sub>2</sub>**

The discrete structure **A** is built from the discrete substructures **b<sub>1</sub>** and **b<sub>2</sub>** of the discrete structure **b[n]** by performing a discrete Inverted Exclusive OR operation.

**A = ROTATE b < n, dir >**

The discrete structure **A** is built by rotating the discrete substructures of the discrete structure **b** over **n** places in the direction **dir**

**A = SHIFT b < n, dir, fill >**

The discrete structure **A** is built by shifting the discrete substructures of the discrete structure **b** over **n** places in the direction **dir**. Empty places are filled with the discrete structure **fill**

**A = MIRROR b[n]**

The discrete structure **A** is built from the **n** discrete substructures  $b_1..b_n$  of the discrete structure **b[n]** by placing the discrete substructures  $b_1..b_n$  in the order  $b_n..b_1$

**A = MUX b[n] <i>**

**A = MUX (x, y,...,z) <i>**

The discrete structure **A** is built by multiplexing **n** discrete substructures  $b_1..b_n$ , or  $x..z$ , per group of **i** sequences of DataElements.

**A[n] = DEMUX b <i>**

**A (x, y,...,z) = DEMUX <i>**

The **n** discrete structures  $a_1..a_n$  or  $x..z$  are built from the discrete structure **b** by demultiplexing per group of **i** DataElements

**(A) = INTERLEAVE b <n<sub>1</sub>, n<sub>2</sub>>**

The discrete structure **A** is built from the discrete structure **b** by interleaving the discrete structure **b** with an interleave depth **n<sub>1</sub>** and an interleave factor **n<sub>2</sub>**

**(A) = DEINTERLEAVE b <n<sub>1</sub>, n<sub>2</sub>>**

The discrete structure **A** is built from the discrete structure **b** by de-interleaving the discrete structure **b** with an interleave depth **n<sub>1</sub>** and an interleave factor **n<sub>2</sub>**

**A = STUFF b <n, {INV}> {recursive}**

The discrete structure **A** is built from the **n** discrete substructures **b**, by adding an (n+1)<sup>st</sup> discrete substructure **b** (optionally inverted), only if the **n** instances of the discrete substructure **b** are identical. Optionally this process is recursive

**A = DESTUFF b <n, {INV}> {recursive}**

The discrete structure **A** is built from the **n+1** discrete substructures **b**, by deleting the (n+1)<sup>st</sup> discrete substructure **b**, only if the **n** instances of the discrete substructure **b** are identical. Optionally this process is recursive

**A = CODE b <@Table>**

the discrete structure **A** is built from the discrete structure **b** using the coding table **Table**

**A = SUBST b <b<sub>1</sub>= 'a<sub>1</sub>', b<sub>2</sub>= 'a<sub>2</sub>', ..., b<sub>n</sub>= 'a<sub>n</sub>'>**

**A = SUBST b <@Table>**

The discrete structure **A** is built from the discrete structure **b** by substitution of the discrete structure 'a<sub>n</sub>' for the instance 'b<sub>n</sub>' of the discrete structure **b** according to the specification given in <>

**A = COUNT b <n, field>**

The discrete structure **A** is built from only those instances of the discrete structure **b** which contain exactly **n** identical sequences of DataElements **field**

**(A, B) = PAR a <1, Odd, lEven>**

**(A, B) = PAR a <n, @Matrix>**

The discrete structure **(A,B)** is built from the discrete structure **a** by performing the operation defined by the matrix **Matrix**

**(A) = CIPHER a <Polynomial, Offset>**

The discrete structure **(A)** is built from the discrete structure **a** by performing the operation of a LFSR with feedback polynomial **Polynomial**

## **Bijlage B      Recommendations Software Specifications for BIAS**

De beschrijving van de RSS volgt op de volgende pagina's en kan beschouwd worden als een zelfstandig onderdeel binnen dit document.

# Recommendation Software Specification voor BIAS

Document identificatie 6426405.000/RSS001  
Contract nr. 861.3886.3963.11  
Versie 1.0  
Datum 15 September 1997

Gerealiseerd door TNO Fysisch en Elektronisch Laboratorium  
Postbus 96864  
2509 JG Den Haag  
Nederland

Gerealiseerd voor DMKL/INFO  
Telecom- en Informatiesystemen  
2597 PC Den Haag  
Nederland

## Goedgekeurd door klant

Functie  
Naam/initialen  
Datum

Functie  
Naam/initialen  
Datum

Functie  
Naam/initialen  
Datum

## Goedgekeurd door TNO-FEL

Functie  
Naam/initialen  
Datum

Functie  
Naam/initialen  
Datum

Functie  
Naam/initialen  
Datum

The classification designation Ongerubriceerd is equivalent to Unclassified, Stg.  
Confidentieel is equivalent to Confidential and Stg. Geheim is equivalent to Secret.

All rights reserved. No part of this publication may be reproduced and/or published by print, photoprint, microfilm or any other means without the previous written consent of TNO.

## Configuratie item historie beschrijving

Versie	Datum	Hoofdstuk	Beschrijving
0.1	01-08-97		Eerste opzet
1.0	21-08-97		Laatste versie

**Inhoud**

1. Inleiding.....1

2. Referenties.....2

3. Mogelijkheden voor software.....3

    3.1 Nieuwe software .....3

    3.2 Aanpassen van software.....3

    3.3 Eindaanbeveling.....4



## 1. Inleiding

Dit rapport beschrijft de aanbevelingen voor de Software Specifications voor 'BIt Analyse Systeem (BIAS)'. Aan de hand van deze aanbeveling wordt een beeld geschetst van de mogelijkheden die er zijn om te komen tot een softwarepakket voor een analysesysteem ten behoeve van de analyse van digitale overdrachtsystemen.

Dit rapport kan echter geen eenduidige uitspraak doen over de uiteindelijke softwareoplossing, maar kan alleen een leidraad zijn voor de te volgen weg in de ontwikkelingen om te komen tot een nieuw analysesysteem. De parameters welke de precieze keuze bepalen zijn hoofdzakelijk:

- kosten,
- koppeling met bestaande systemen,
- capaciteit en
- de beschikbare tijd.

Aangezien geen van de vier randvoorwaarden nu al vastliggen is een kant-en-klare oplossing niet te geven.

De Recommendations Software Specifications zijn een gevolg van eerder uitgevoerde onderdelen binnen het project BIAS. Ten behoeve van de aanbevelingen is een Software Requirement Specification (SRS) opgesteld.

## 2. Referenties

In de tekst is gerefereerd naar :

Document identificatie	Document naam	Versie	Datum	Gepubliceerd door
SRS_103	Software Requirement Specifications for BIAS	2.0	22-08-'97	TNO-FEL

### 3. Mogelijkheden voor software

De software-randvoorwaarden voor een softwarepakket ten behoeve van BIAS liggen nog niet vast. Deze bestaan uit:

- het type platform,
- het Operating System,
- de te gebruiken GUI en
- de interfaces naar apparatuur en andere softwarepakketten.

Andere software is aanwezig, echter de kwantiteit en kwaliteit is niet hoog. Daarom kan er voor BIAS gekozen worden uit een tweetal oplossingen:

- de bouw van nieuwe software en
- een aangepaste versie vervaardigen.

#### 3.1 Nieuwe software

Bij het schrijven van nieuwe software, kunnen alle randvoorwaarden nog vrij gekozen worden. Het opnieuw schrijven van software heeft de volgende voor en nadelen:

- + voldoet optimaal aan de gestelde SRS,
- + keuze van platform is vrij,
- + keuze GUI is vrij, echter een window-like GUI,
- + onafhankelijk van omgeving,
- + geen historie.
- hoge kosten,
- ontwikkelrisico is hoog,
- interface naar andere in ontwikkeling zijnde apparatuur variabel,
- mogelijke 'kinderziektes'.

#### 3.2 Aanpassen van software

De software moet worden aangepast en/of uitgebreid om tegemoet te komen aan de eisen zoals deze staan vermeld in de SRS[1].

Het aanpassen van software heeft ook zijn voor en nadelen;

- + relatief lage kosten,
- + ontwikkelrisico is laag,
- + kinderziektes uit 'oude' software kunnen dan ook verholpen worden,
- + er is een gebruikersgroep aanwezig,
- + interface reeds aanwezig.
- platform is al gedefinieerd,
- GUI staat al vast,
- resultaat kan uitlopen op een compromis,
- problemen met werkomgeving,
- software bevat historie.

### 3.3 Eindaanbeveling

Indien één van de randvoorwaarden, de kosten, buiten beschouwing wordt gelaten zijn de enige bezwaarlijke redenen tegen het niet schrijven van nieuwe software; het mogelijk langere ontwikkeltraject en het feit dat er hoogstwaarschijnlijk gebruik gemaakt gaat worden van een andere aannemer. Het grootste voordeel van nieuwe software is het feit dat deze zodanig geschreven zal worden dat ook in de verdere toekomst nog de software voldoet aan de eisen gesteld in de SRS. De ervaring leert dat aanpassing van oude software toch altijd consessies doet aan de vooraf gestelde eisen, in dit geval de SRS[1].

**REPORT DOCUMENTATION PAGE**  
**(MOD-NL)**

1. DEFENCE REPORT NO (MOD-NL) TD97-0154	2. RECIPIENT'S ACCESSION NO	3. PERFORMING ORGANIZATION REPORT NO FEL-97-A253
4. PROJECT/TASK/WORK UNIT NO 6026405	5. CONTRACT NO A97KL790	6. REPORT DATE September 1997
7. NUMBER OF PAGES 76 (incl 2 appendices, excl RDP & distribution list)	8. NUMBER OF REFERENCES -	9. TYPE OF REPORT AND DATES COVERED
10. TITLE AND SUBTITLE Software Specificaties BIAS (Software Specifications BIAS)		
11. AUTHOR(S) H.G.M. Bennis A.P.T.M. Onderwater		
12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) TNO Physics and Electronics Laboratory, PO Box 96864, 2509 JG The Hague, The Netherlands Oude Waalsdorperweg 63, The Hague, The Netherlands		
13. SPONSORING AGENCY NAME(S) AND ADDRESS(ES) DMKL/INFO/Telecom- en Informatiesystemen Van der Burchlaan 31, 2597 PC Den Haag, The Netherlands		
14. SUPPLEMENTARY NOTES The classification designation Ongerubriceerd is equivalent to Unclassified, Stg. Confidentieel is equivalent to Confidential and Stg. Geheim is equivalent to Secret.		
15. ABSTRACT (MAXIMUM 200 WORDS (1044 BYTE)) This document describes the Software Requirement Specifications(SRS) and the Recommendations Software Specifications(RSS) for a BIIt Analysis System(BIAS) for the Royal Dutch Army. This BIAS is a system to analyse intercepted long haul digital communications links. The SRS is developed according to MIL-STD-498. The resulting SRS shall be used to define and implement a BIAS or redefine an existing analysis program.		
16. DESCRIPTORS Military Communications Wireless Communications		IDENTIFIERS Digital signal analysis Digital data communication
17a. SECURITY CLASSIFICATION (OF REPORT) Ongerubriceerd	17b. SECURITY CLASSIFICATION (OF PAGE) Ongerubriceerd	17c. SECURITY CLASSIFICATION (OF ABSTRACT) Ongerubriceerd
18. DISTRIBUTION AVAILABILITY STATEMENT Unlimited Distribution		17d. SECURITY CLASSIFICATION (OF TITLES) Ongerubriceerd

## Distributielijst

1. Bureau TNO Defensieonderzoek
2. Directeur Wetenschappelijk Onderzoek en Ontwikkeling\*)
3. HWO-KL
4. HWO-KLu\*)
5. HWO-KM\*)
6. HWO-CO\*)
- 7 t/m 9. KMA, Bibliotheek
- 10 t/m 20. DMKL/INFO Telecom- en Informatiesystemen, t.a.v. W.A.J. Egberts
21. Directie TNO-FEL, t.a.v. Dr. J.W. Maas
22. Directie TNO-FEL, t.a.v. Ir. J.A. Vogel, daarna reserve
23. Archief TNO-FEL, in bruikleen aan M&P\*)
24. Archief TNO-FEL, in bruikleen aan Ir. H.G.M. Bennis
25. Archief TNO-FEL, in bruikleen aan Ir. A.P.T.M. Onderwater
26. Documentatie TNO-FEL
- 27 t/m 30. Reserve

TNO-PML, Bibliotheek\*\*)

TNO-TM, Bibliotheek\*\*)

TNO-FEL, Bibliotheek\*\*)

Indien binnen de krijgsmacht extra exemplaren van dit rapport worden gewenst door personen of instanties die niet op de verzendlijst voorkomen, dan dienen deze aangevraagd te worden bij het betreffende Hoofd Wetenschappelijk Onderzoek of, indien het een K-opdracht betreft, bij de Directeur Wetenschappelijk Onderzoek en Ontwikkeling.

\*) Beperkt rapport (titelblad, managementuittreksel, RDP en distributielijst).

\*\*) RDP.